THE UNIVERSITY OF HONG KONG

DEPARTMENT OF MECHANICAL ENGINEERING

MECH 4429 | CAES 9544

Integrated Capstone Experience

Final Report

Project Title:	Distributed Formation Control of A Swarm of UGVs
Student Name:	Chan Hing Cheung (3035276452) Chiang Hou Nam (3035299260) Leung King Tsung (3035281902) Tsoi Wai Lok (3035275733) Wong Ho Wai (3035279181)
Degree Programme:	BEng(ME)
Supervisor:	Prof. Lam J
Moderator:	Dr. Kwok KW
Date:	8th May, 2020

Abstract

Multi-agent Systems (MAS) have become more common in recent years. Common MAS include Unmanned Ground Vehicles (UGVs) [1, 2], Unmanned Aerial Vehicles [3], Satellites [4] and Autonomous Underwater Vehicles [5]. However, controlling MAS has been an issue in that centralized control algorithms cannot be extended to large scale due to limitations of computational power, communication bandwidth and distances [6]. Thus, as a form of cooperative control, distributed formation control algorithms are introduced to solve the complex problems by maintaining a given geometrical shape in the MAS. Distributed formation control algorithms are extendable to a swarm of UGVs. The objectives in this study include investigating path navigation of single UGV, different formation and consensus algorithms commonly adopted in researches and performing distributed formation control of multiple UGVs with time-varying formation patterns. The feasibility and performance of a virtual leader approach of formation control scheme using a low cost distributed computational system is evaluated. Results show 7 good formations using a consensus algorithm along with directions for future work which aims to improve the robustness of the formation. The ability of controlling UGVs in distributed manner is essential in the near future as autonomous vehicles are becoming more common. Furthermore, the practical application of distributed formation control includes but not limited to search and rescue after disasters, weather prediction and underwater contour mapping. Thus, the study of distributed formation control is valuable and has significant societal implications.

Acknowledgements

We would like to pay our special regards to our supervisor, Prof. Lam J, for his clear guidelines of the project and thorough explanation of literatures. Also, we would like to thank Alex (Xin Gong), a PhD student of Prof. Lam, for leading us to configure ROS and Ubuntu OS. Without them, the configuration process and understanding of literatures would be greatly hindered.

We wish to express our deepest gratitude to our CAES teacher Mable for outlining the contents of the progress report with details. The outlines are providing distinct guidance to report writing. The reviews of the Interim Report also pointed out our misunderstanding of sections of report, driving us to complete this report with clear objectives.

We would like to thank the technicians in the Lab, especially Mr. Lee, for assisting us during the configuration of OptiTrack system. We would like to recognize the invaluable assistance that you all provided during our study.

Statement of Contributions

Chan Hing Cheung has done the following contributions in this study:

1) Kinematics and position control of differential drive robot
2) Simulation of control of differential drive robot
3) Path tracking of differential drive robot

Chiang Hou Nam has contributed the following parts in this study:

1) Literature Review of consensus and formation control algorithms
2) Consensus and formation control module
3) Navigation of multiple robots

Leung King Tsung contributed following parts in this study:

1) Communication between master PC and robots
2) Path planning algorithm study for multiple robots
3) Consensus and formation control module

Tsoi Wai Lok has contributed the following parts in this study:

1) Literature Review of graph theory and consensus algorithms
2) Simulation of control of differential drive robot
3) Kinematics of differential drive robot

Wong Ho Wai has done the following contributions in this study:

1) Communication between master PC and robots
2) Communication between master PC and OptiTrack Motion Capture System
3) Navigation of multiple robots

Table of Contents

Abstract.....*i*

Acknowledgements..... *ii*

Statement of Contributions..... *iii*

List of Figures.....*viii*

List of Tables.....*xi*

Abbreviations.....*xii*

Notations.....*xiii*

CHAPTER 1: INTRODUCTION.....1

1.1 Background and Motivation.....1

1.2 Definition of Formation Control.....2

1.3 Objectives of the Study.....3

1.4 Significance of the Study.....3

1.5 Outline of the Report.....4

CHAPTER 2: LITERATURE REVIEW.....5

2.1 Introduction.....5

2.2 Consensus Algorithms.....5

2.2.1 Graph Theory.....5

2.2.2 Consensus Algorithms.....7

2.3 Formation Control Algorithms.....9

2.4 Summary.....10

CHAPTER 3 : METHODOLOGY 12
3.1 Introduction 12
3.2 Experimental Setup 12
3.2.1 Software Environment 13
3.2.2 TurtleBot3 14
3.2.3 OptiTrack Motion Capture System 17
3.2.4 Setup Summary 20
3.3 TB3 Control Algorithms 21
3.3.1 Kinematics of Differential Drive Robots 21
3.3.2 Position Control of Differential Driven Robots 23
3.3.3 Path Planning 25
3.3.4 Simulation Tests 35
3.3.5 Navigation Tests 36
3.3.6 Communication of Multiple Robots 37
3.4 Design of Formation Control Scheme 38
3.5 Summary 43

CHAPTER 4: RESULTS AND DISCUSSION 44
4.1 Introduction 44
4.2 Robot Navigation Results 44
4.2.1 Path Tracking Results of Single Robot 44
4.2.2 Path Planning Results of Single Robot 49
4.3 Distributed Formation Control Results 51
4.3.1 Formation One: Circle Formation 52
4.3.2 Formation Two: Line Formation One 56

4.3.3 Formation Three: Line Formation Two 60
4.3.4 Formation Four: Rotating Circle Formation 64
4.3.5 Formation Five: Rotating Line Formation 68
4.3.6 Formation Six: Spiral Formation 72
4.3.7 Formation Seven: Flower Formation 76
4.4 Limitations and Difficulties Encountered 80
4.4.1 OptiTrack: Missing Packages 80
4.4.2 Varying Control Frequency in ROS 80
4.4.3 Difficulties in real-time Obstacle avoidance 81
4.5 Summary 81

CHAPTER 5: CONCLUSION 82
5.1 Objectives Recap 82
5.2 Major Findings 82
5.3 Conclusions 82

CHAPTER 6: FUTURE WORK 84
6.1 Pose Estimations 84
6.2 Real-time Obstacle Avoidance 84
6.3 Acceleration Limits 85
6.4 Time-variant Update Rate of Formation 85
6.5 Motive Updates 85
6.6 Time-variant Communication Network of Formation 86
6.7 Formation rearrangement 86

REFERENCES.....*87*

List of Figures

Figure 1.1: Space interferometry with the use of multiple spacecraft 1
Figure 1.2: Flock of birds in formation 2

Figure 2.1: An undirected graph $G = N, E$ 6
Figure 2.2: Virtual leader approach with a virtual leader and four followers 10

Figure 3.1: Schematic diagram of the information flow 12
Figure 3.2: Isometric view of a TurtleBot3 Burger 14
Figure 3.3: Topic in ROS Communication Network 15
Figure 3.4: Setup of OptiTrack motion capture system indoor 17
Figure 3.5: A Prime 13 motion camera (Left) and the setup in laboratory (Right) 17
Figure 3.6: Connection of Optitrack Motion Capture System 18
Figure 3.7: Spherical reflective marker mounted on the robot in different pattern 18
Figure 3.8: OptiTrack Motion Capture Software (Motive 1.8) information 19
Figure 3.9: Flow Chart of Data Streaming from OptiTrack software to ROS network 20
Figure 3.10: Demonstration of TB3 Burger with $\xi = x\ y\ \theta\ T$ 21
Figure 3.11: Demonstration of TB3 Burger moving from current pose to goal pose 23
Figure 3.12: Control block diagram of single robot dynamics 25
Figure 3.13: Simulation of Quintic Polynomial Path planning 28
Figure 3.14: Flow chart of Quintic Polynomial Path Planning 29
Figure 3.15: Heuristic distance and actual distance in graph calculation 30
Figure 3.16: Searching Simulation of Dijkstra's algorithm 31
Figure 3.17: Searching Simulation of Greedy Best-First-Search algorithm 32
Figure 3.18: Searching Simulation of A* algorithm 33
Figure 3.19: Example of Potential Field Algorithm 34
Figure 3.20: Block diagram of position control in Simulink 35
Figure 3.21: Block diagram of differential drive kinematics in Simulink 35
Figure 3.22: Block diagram of differential drive kinematics with disturbance in Simulink 36

Figure 3.23: Communication diagram between master PC and the robots in ROS system 37
Figure 3.24: A formation composed of four agents with a consistent (right) and inconsistent (left) understanding of a formation frame 38
Figure 3.25: Distributed formation control architecture using a virtual leader approach 41

Figure 4.1: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different $k\rho$ 45
Figure 4.2: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different $k\alpha$ 46
Figure 4.3: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different $k\beta$ 46
Figure 4.4: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different disturbances.....47
Figure 4.5: Path tracking of TB3 Burger with initial pose and goal pose 48
Figure 4.6: Path planning of turtlebot with Quintic Polynomials planner.....49
Figure 4.7: Path planning of turtlebot with A* Path Planning Algorithm.....50
Figure 4.8: Obstacle avoidance result with potential field algorithm - Case one.....50
Figure 4.9: Obstacle avoidance result with potential field algorithm - Case two.....51
Figure 4.10: Communication network of Formation one 52
Figure 4.11: Expected robot locations of Formation 1 53
Figure 4.12: Actual Paths of individual robots of Formation one 54
Figure 4.13: Positional Error of Individual robot in Formation one.....55
Figure 4.14: Communication network of Formation two.....56
Figure 4.15: Expected robot locations in Formation two 57
Figure 4.16: Actual Paths of individual robots of Formation two.....58
Figure 4.17: Positional Error of Individual robot in Formation two 59
Figure 4.18: Communication network of Formation three.....60
Figure 4.19: Expected robot locations in Formation three 61
Figure 4.20: Actual Paths of individual robots of Formation three.....62
Figure 4.21: Positional Error of Individual robot in Formation three 63
Figure 4.22: Communication network of Formation four 64
Figure 4.23: Expected robot locations in Formation four.....65
Figure 4.24: Actual Paths of individual robots of Formation four 66
Figure 4.25: Positional Error of Individual robot in Formation four.....67
Figure 4.26: Communication network of Formation five.....68

Figure 4.27: Expected robot locations in Formation five 69
Figure 4.28: Actual Paths of individual robots of Formation five 70
Figure 4.29: Positional Error of Individual robot in Formation five 71
Figure 4.30: Communication network of Formation six 72
Figure 4.31: Expected robot locations in Formation six 73
Figure 4.32: Actual Paths of individual robots of Formation six 74
Figure 4.33: Positional Error of Individual robot in Formation six 75
Figure 4.34: Communication network of Formation seven 76
Figure 4.35: Expected robot locations in Formation seven 77
Figure 4.36: Actual Paths of individual robots of Formation seven 78
Figure 4.37: Positional Error of Individual robot in Formation seven 79

List of Tables

Table 3.1: Hardware specifications of Turtlebot3 14
Table 3.2: Common terms and descriptions in ROS..... 16
Table 3.3: Simultaneous Equation of Quintic Polynomials Algorithm 26

Table 4.1: Values of kinematics parameters in MATLAB Simulation 44
Table 4.2: Motion times of robot from$(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different $k\rho$ 45
Table 4.3: Values of kinematics of TB3 Burger 48

Abbreviations

FPS	Frames Per Second
GUI	Graphic User Interface
IMU	Inertial Measurement Unit
LAN	Local Area Network
MAS	Multi-agent Systems
OS	Operating System
PC	Computer
PID	Proportional-Integral-Derivative
ROS	Robot Operating System
TB3	TurtleBot3
UAV	Unmanned Aerial Vehicles
Ubuntu OS	Ubuntu Operating System
UGV	Unmanned Ground Vehicle
VL	Virtual Leader

Notations

$\boldsymbol{A}$	Adjacency matrix
a_{ij}	Entries in adjacency matrix at the ith row and the jth column
C_F	Formation coordinate
C_{Fj}	Agent j's understanding of C_F
$\boldsymbol{D}$	Degree matrix
d_{ij}	Entries in degree matrix at the ith row and the jth column
E	Number of edges in graph theory context
$\overrightarrow{e_i}$	Input for PI Controller for agent i
G	A Graph $G = (N, E)$: A graph with N nodes and E edges
K_P	Positional tuning parameter in PI Controller
K_I	Integrative tuning parameter in PI Controller
k_ρ	Tuning Parameter for ρ
k_α	Tuning Parameter for α
k_β	Tuning Parameter for β
N	Number of nodes in graph theory context
$\boldsymbol{L}$	Laplacian matrix
L_w	Distance between the wheels of a robot
$\mathbb{R}^{m\times n}$	A matrix with $m \times n$ components will all of them are real numbers
$\vec{r}_i$	Position vector of agent i in terms of $[x_i \quad y_i \quad \theta_i]^T$
$\vec{r}_j^{\,d}$	Desired position of agent j in terms of $\begin{bmatrix} x_j^d & y_j^d \end{bmatrix}^T$
$\vec{r}_{jF}$	Position vector of agent j reference to the formation frame in terms of $[x_{jF} \quad y_{jF}]^T$
r_L	Radius of the left wheel of a robot
r_R	Radius of the right wheel of a robot
$\vec{u}_i$	Output from PI Controller / Output from the Consensus-based Formation Control Module
v	Linear velocity
x	Dynamics: position at x direction

x_B	Path Tracking: Starting x position
x_d	x coordinate of the desired formation state
$x_{d,j}$	The understanding of the x coordinate of C_{Fj} of agent j
x_i	Consensus: Information state of agent i
$\dot{x}_i$	Consensus: First derivative of information state $x_i(t)$ / Dynamics: velocity at x direction of agent i
x_{jF}	x coordinate of agent j reference to the formation frame
y	Dynamics: position at y direction
y_B	Path Tracking: Starting y position
y_d	y coordinate of the desired formation state
$y_{d,j}$	The understanding of the y coordinate of C_{Fj} of agent j
y_{jF}	y coordinate of agent j reference to the formation frame
$\dot{y}$	Dynamics: velocity at y direction
α	Steering angle
β	Path Tracking: goal orientation
Δ_x	Positional difference between the start pose and goal pose in x direction
Δ_y	Positional difference between the start pose and goal pose in y direction
$\varkappa$	A tuning parameter in formation control
ω	Angular velocity of a robot
ω_L	Angular velocity of the left wheel
ω_R	Angular velocity of the right wheel
ρ	Path Tracking: Distance between the start pose and the goal pose
θ	Robot orientation
$\dot{\theta}$	Angular velocity of a robot
θ_d	Orientation of the desired formation state
$\theta_{d,j}$	The understanding of the orientation of C_{Fj} of agent j
$\vec{\xi}_d$	Desired formation state at C_F in terms of $[x_d \quad y_d \quad \theta_d]^T$
$\vec{\xi}_j$	The understanding of the coordinate C_{Fj} of agent j in terms of $[x_{d,j} \quad y_{d,j} \quad \theta_{d,j}]^T$
$\in$	Belongs to

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Formation control of multi-agent systems (MAS) has been significantly researched in recent years to meet the demand of autonomous control of multiple agents, including Unmanned Ground Vehicles (UGVs) [1, 2], Unmanned Aerial Vehicles [3], Satellites [4] and Autonomous Underwater Vehicles [5]. In the past, most MAS were constructed based on centralized control protocols [7], leading to the requirement of communication between all agents and large computation loads. Distributed formation control requires communication links between neighbors instead of all agents and distributes computation loads to agents [6], providing larger flexibility and scalability of the MAS.

Formation control of MAS promises a wide range of practical applications. Road safety could be enhanced by eliminating human error and efficiency in transportation could be increased with UGVs [8]. Search and rescue after disasters, such as earthquakes and wildfire, could be performed by Unmanned Aerial Vehicles formation. Weather prediction and the Universe exploration could be provided by satellite formation flying. These applications provide impetus for this study to focus on the distributed formation control of multi-agent systems especially the field of UGVs.

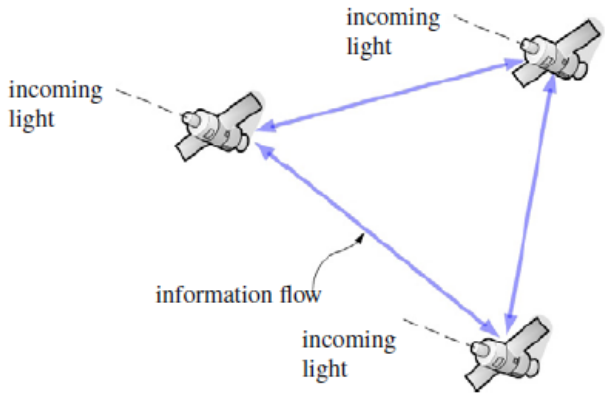

Figure 1.1: Space interferometry with the use of multiple spacecraft

In space interferometry (Figure 1.1), spacecraft must hold precise relative distances at the same level towards the incoming light to produce the correct interferometry pattern. The relative distances between the spacecraft are measured using the optical sensors which use to exchange information between the spacecraft [9].

1.2 Definition of Formation Control

Formation control is categorized in the cooperative control family, which consists of a group of agents with the ability of sensing and/or communication. The common goal of these agents in the context of cooperative control is to achieve assigned individual agent and group behaviors using only relative information such as relative positions and velocities. This relative information is obtained through an information flow architecture between agents which is dictated by the relative sensing and communication. Therefore, a cooperative control system can be seen as consisting of four elements:

1) Agents
2) Information topology
3) Motion control algorithms
4) Group objective

Examples of such group objectives include duties as guarding and escorting.

Specifically, the objective of the formation control is to stabilize relative distances between the agents to prescribed values [9]. Moreover, this phenomenon has occurred casually in the animal world like birds (Figure 1.2) and fish [10].

Figure 1.2: Flock of birds in formation

In this study, the distributed formation control for a swarm of UGVs will be investigated and implemented.

1.3 Objectives of the Study

The study aims to use distributed formation control of UGVs to provide a foundation of large-scaled autonomous ground vehicles. The objectives of the study are:

(1) To investigate and perform navigation of single UGV with path planning, path tracking and obstacle avoidance.
(2) To investigate different formation and consensus algorithms commonly adopted in research studies.
(3) To perform distributed formation control of multiple UGVs with time-varying formation patterns.
(4) To evaluate the feasibility and performance of a virtual leader approach of formation control scheme using a low cost distributed computational system.

1.4 Significance of the Study

With the extensive research in the nonlinear control system for swarm robot, the demand for computational power has increased exponentially. While the swarm robots control systems are generally enforceable, distributed control subsystem approach is becoming feasible in the industry. Together with the rise of trajectory tracking systems, a low cost distributed computational system for swarm robot is in demand. The study will focus on the distributed formation control of UGVs. Four UGVs will be used in this study. The control algorithms are not limited to small scale UGVs. They could be extended to a swarm of UGVs which will be a common way of transportation in the foreseeable future. In addition, the control algorithms of formation of UGVs could also be applied to other MAS, such as Unmanned Aerial Vehicles, Satellites and Autonomous Underwater Vehicles. It is clear that a variety of applications are rendered possible by the formation of different MAS.

Since there are great research interests on proposing formation control system in a theoretical manner [11] [12] [13] [14] [15] [16] [17] [18], the main contribution of this study lies in testing the feasibility and performance of such system in the real world and investigate the difficulties and limitations of implementing such system.

1.5 Outline of the Report

Chapter 2 introduces graph theory and consensus algorithms which are techniques for agents in MAS to reach common goals. Then common formation control approaches towards MAS will be investigated with the selection of appropriate formation control algorithm.

Chapter 3 presents the environment setups of the experiments, including hardware and software setups. Control algorithms of a brand of UGVs, TurtleBot3 (TB3), are then discussed with the corresponding tests to validate them. In addition, distributed formation control architecture is included.

Chapter 4 shows the navigation results of TurtleBot3 Burger, including results of path planning, path tracking and the combination of path planning and tracking. Formation results are also presented with seven different formations. Moreover, limitations and difficulties of the study are discussed in this chapter.

Chapter 5 concludes the significant results in this study. The contributions of this study are also enclosed in this Chapter.

Chapter 6 introduces future works including feedback estimations, real-time obstacle avoidance, acceleration limits and time-variant update rate of formation which could solve the inadequacies and limitations of this study.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, a review on the topic of distributed formation control is given. An introduction and the mathematics related to consensus algorithms with the proof of convergence will be given in part 1. Then, an introduction of the formation control and justification of the approach used in this study will be given in part 2.

2.2 Consensus Algorithms

Consensus, in layman's terms, means agreement. Analogically, in terms of cooperative control, consensus illustrates the phenomenon when multiple agents have consistent values of a shared variable of interest, which is regarded as the information state. To reach consensus, suitable methods are required to converge that information state to the destined value. These methods are called consensus algorithms. Since "agreement" should only exist groupwise, it is reasonable to assume neighbor to neighbor only interaction between agents. Therefore, consensus algorithms are essentially update principles to converge the information states of all the agents in the network to a common value [19].

Since a network is involved in this area, relevant areas in graph theory are also reviewed and summarized.

2.2.1 Graph Theory

An undirected graph $G = (N, E)$ represents a communication network among n independent agents where $N = \{1,2, \dots, n\}$ is a set of nodes without empty entry, and $E = (i, j) \in N \times N$ is a set of undirected edges. Undirected graphs can be used to model a network for communication among n independent agents. In the graph, each node can be modelled as an agent and each edge $E = (i, j)$ can be modelled as the communication between agent i and j. Agents i and j are considered as neighbors if and only if $E = (i, j)$ exists. In this study, self-edges $E = (i, i)$ are prohibited because of the meaningless and inefficient communication within the same individual when having cooperation. The graph G is regarded as connected if there is a path between every pair of nodes [20].

A mathematical equivalent expression to represent the graph G is its adjacency matrix A:

$$\boldsymbol{A} = [a_{ij}] \in \mathbb{R}^{n \times n}$$
$$a_{ij} = a_{ji} = \begin{cases} 1 & if\ (i,j) \in E \\ 0 & otherwise \end{cases} \tag{2.1}$$

The adjacency matrix is a binary matrix such that every row of the adjacency matrix shows the existence of communication between agent i and other agents. The entry $a_{ij} = 1$ if there is a connection, and $a_{ij} = 0$ otherwise.

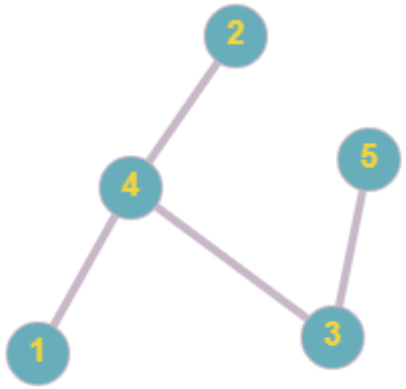

Figure 2.1: An undirected graph $G = (N, E)$

In Figure 2.1, the undirected graph G represents a communication network among five agents. The corresponding adjacency matrix $\boldsymbol{A}$ is given by:

$$\boldsymbol{A} = \boldsymbol{A}^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.2}$$

The diagonal entries are all 0s since self-edges are restricted. Moreover, in a complete graph, every agent in this network can obtain information from all other agents. Since edges exist between every pair of nodes, the adjacency matrix of such a graph will be all 1s except the diagonal entries.

Another matrix which contributes to describe a graph is called the Laplacian matrix L:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \tag{2.3}$$

$\boldsymbol{D}$ here denotes the degree matrix:

$$\boldsymbol{D} = \left[d_{ij}\right] \in \mathbb{R}^{n\times n}$$

$$d_{ij} = \begin{cases} \sum_{j=1}^{n} a_{ij} & i = j \\ 0 & i \neq j \end{cases} \tag{2.4}$$

where the diagonal entries represent the number of agents paired with agent i [20].

The idea of nodes and edges, and the corresponding mathematical representation provide adequate prerequisite knowledge of the consensus algorithm.

2.2.2 Consensus Algorithms

As mentioned in Section 2.2.1, information sharing is crucial when a group of agents is trying to achieve some common goals, and consensus algorithm serves as an update principle that converges the information state of every agent in the network to a certain value. A frequently recognized form of consensus algorithm [21, 22, 23, 24, 25, 26] is found to be in the form of a first-order differential equation:

$$\dot{x}_i(t) = -\sum_{j=1}^{n} a_{ij}(t)\left[x_i(t) - x_j(t)\right], \qquad i = 1, \dots, n \tag{2.5}$$

In this algorithm, $x_i(t)$ denotes the information state of agent i at time t, and $a_{ij}(t)$ corresponds to the entry of the ith row and jth column in the adjacency matrix A. This algorithm can also represent the consensus of the whole group using a matrix form [19, 26]:

$$\dot{\vec{x}}(t) = -\boldsymbol{L}(t)\vec{x}(t) \tag{2.6}$$

where $\vec{x}(t) = [x_1(t), \dots, x_n(t)]^T$ is a vector of information states of all agents and L is the Laplacian matrix. From equation (2.5), it can be seen that if the term $\left[x_i(t) - x_j(t)\right]$ is repetitively worked out for every paired neighbor j in every agent i, the information state of

agent i will negotiate with the information state of its agents j, and eventually drives to a certain value. In this mode of local communication, the information flow is said to be "distributed".

Consensus is reached when at certain instance t:

$$x_1(t) = x_2(t) = \cdots = x_n(t) \tag{2.7}$$

When the condition (2.7) is reached, equation (2.5) will give a null solution such that $\dot{x}_i(t) = 0$ which means there will not be further movement or rate of change of certain attribute.

Lyapunov Stability in the field of Stability Theory [27] is adopted to prove the convergence of this consensus algorithm. For a system of non-linear differential equations:

$$\frac{d\vec{X}}{dt} = f\left(\vec{X}(t)\right) \tag{2.8}$$

Where $\vec{X}(t)$ represents the system state vector with initial conditions $\vec{X}(0) = \vec{X}_0$. Suppose the function f has an equilibrium X_e such that $f(X_e) = 0$, then this equilibrium is regarded as Lyapunov stable, if for every $\varepsilon > 0$, there exists a $\delta > 0$ such that if:

$$\left|\vec{X}(0) - X_e\right| < \delta \tag{2.9}$$

then for all $t \geq 0$:

$$\left|\vec{X}(t) - X_e\right| < \varepsilon \tag{2.10}$$

Furthermore, based on the above, the system of equations is regarded as asymptotically stable if the following is also satisfied:

$$\lim_{t\to\infty}\left|\vec{X}(t) - X_e\right| = 0 \tag{2.11}$$

For the consensus algorithm stated in (2.5), the negative sign is noted that:

$$\dot{x}_i(t) < 0 \tag{2.12}$$

Therefore, $x_i(t)$ must be a decreasing function and since the goal of the consensus algorithm is to agree on a value among the neighbours, it is reasonable to regard $x_j(t)$ as the equilibrium of $\dot{x}_i(t)$, which corresponds to x_e in Lyapunov stability theorem.

Initially at time $t = 0$, there will be a notable difference k between the information state of agents i and j, such that:

$$\left|x_i(0) - x_j(0)\right| = k < \delta \tag{2.13}$$

Subsequently, the difference between the information state of agents i and j will decrease and eventually:

$$\left|\overrightarrow{x_i}(t) - \overrightarrow{x_j}(t)\right| \to 0 \tag{2.14}$$

which satisfies asymptotic Lyapunov stability.

2.3 Formation Control Algorithms

Studies in the area of formation control abound, some of them are investigated in [21, 28, 29, 30, 31, 32, 33]. Specifically, there has been great research interest in the application of the formation of multiple UGVs such as [11, 16, 17, 19, 21, 28, 33].

There are several approaches and strategies proposed for formation control and these can be categorized as:

(1) Leader following [34, 35]
(2) Behavioral [36]
(3) Virtual leader [37, 38]
(4) Graph rigidity [39, 40]
(5) Artificial potential functions [41]

In these approaches, it is found that the virtual leader approach would be the best fit for the purposes of this study which could be achieved in three steps [42]:

(1) Defining desired dynamics for the virtual leader
(2) Translating the motion dynamics of the virtual leader to the motions of each agent
(3) Deriving the tracking controls for each agent towards the formation

This method would best suit the study because a point in the formation, usually the virtual center, as the virtual leader, can be modeled. The formation system can be moved around if the virtual center (leader) is allowed to move. And most importantly, since this approach allows only neighbor to neighbor communication, “Distributed” requirement in the study is satisfied.

Figure 2.3 demonstrates an example of virtual leader approach of UGVs with a virtual leader and four followers. Only the agents not following other agents are considered as leaders. Apart from UGVs, the use of virtual leader approach has also been used in some applications which demand high precision such as the formation of spacecraft in free space [43, 44]. Therefore, this approach can be regarded as the one designated for high accuracy applications.

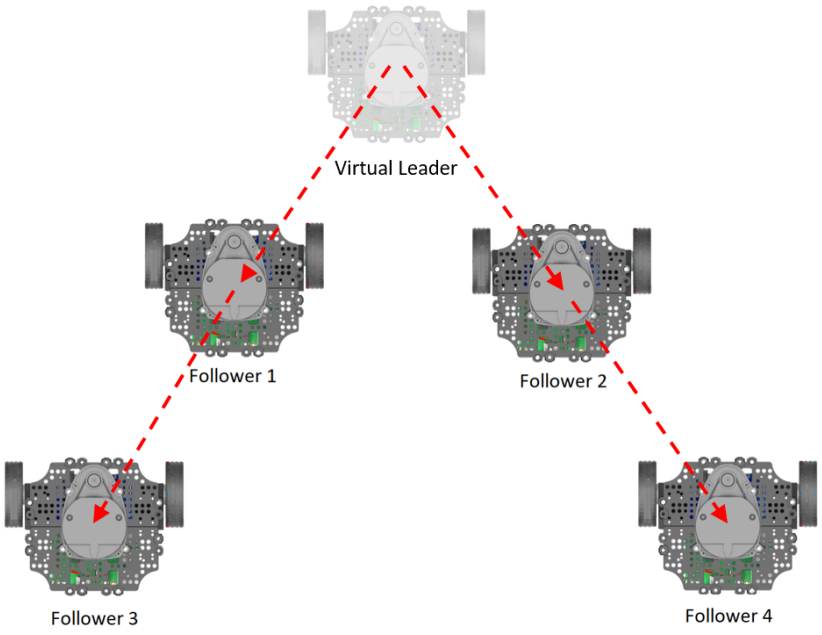

Figure 2.2: Virtual leader approach with a virtual leader and four followers

Other approaches have also been explored, but some shortcomings and significant knowledge gaps are observed. For example, the behavioral approach involves weight averaging the control of several desired behaviors prone to unpredictable and strange behaviors because these behaviors compete among themselves and are averaged [45]. Graph rigidity and artificial potential functions require relatively advanced knowledge such as game theory and Nash equilibrium [46].

In chapter 3, the methodology for the formation used will be derived and the proof that formation control can be a consensus problem will be shown.

2.4 Summary

In this chapter, the concept of consensus is reviewed, relevant knowledge of graph theory is introduced, the convergence of the consensus algorithm is also proved and the approaches and

the selection of formation control are addressed. In the next chapter, the methodology and approach of the study will be discussed.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the methodology of this study. The first part of this chapter will give an overview of the setup of the study. Next, the mathematical model for the robot dynamics and control will be derived. Lastly, the derivation of the mathematics for the formation control will be given.

3.2 Experimental Setup

The schematic diagram of this study is first introduced. Figure 3.1 shows a schematic diagram of the study. The position and angular orientations of the robots can be detected by the OptiTrack motion capture system. Then, the master PC will be able to get the information of identical robots from the communication network, and send the calculated formation result to the leader agents. The robots will then share positional information among themselves and then agree on a common ground, which is consensus, and form the formation based on the consensus result.

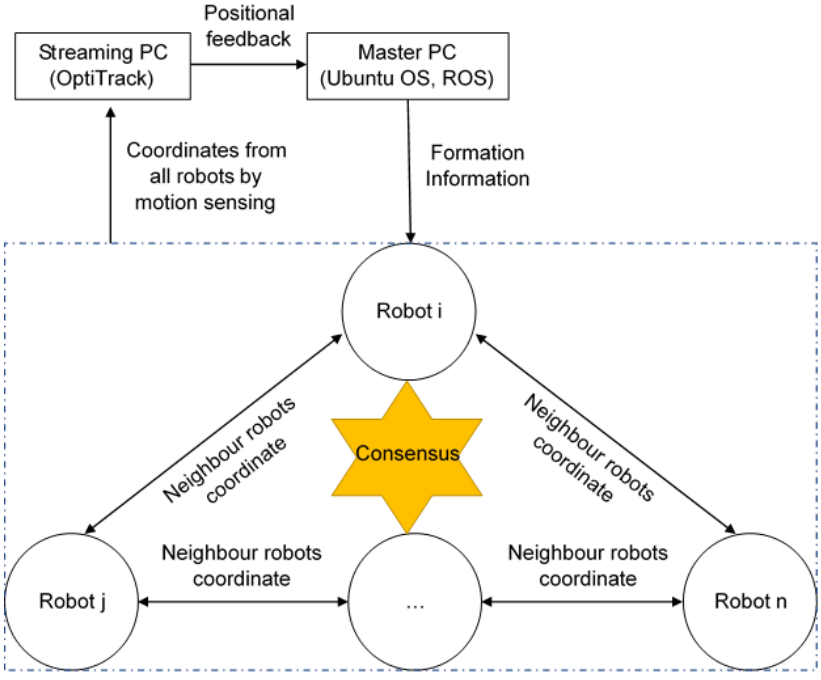

Figure 3.1: Schematic diagram of the information flow

3.2.1 Software Environment

3.2.3.1 ROS kinetic ubuntu 16.04

ROS is an open-source operating system which is designed for robot systems. Different functions and libraries are provided for the system, including hardware abstraction, low-level device control, data passing between different processes, and package management. ROS also provides tools and libraries for writing and running code by different languages, such as python and C++.

In the study of distributed formation control, as the robot system needed to control multiple turtlebots with one single master computer, ROS will be a convenient system to manage and communicate between both Turtlebot3, OptiTrack Motion Capture system and the master computer.

There are also several graphical user interfaces (GUI) in ROS that allow the user to manually control topic values or monitor variables in a particular set of code. RViz and rqt were used to monitor topics instead of editing the raw code and let the code print text on the terminal.

3.2.3.2 Python

Python is the programming language used in this study, not only is it used by many developers and related projects that have been referred to in this study, but also the abundance of packages Python contain to visualize the performance of the controller before publishing the code on the robots themselves. Plenty of codes made referenced to in this study also require Python packages and different distributions, and effort expended on transposing the algorithms to the result is also done on Python.

Being a high-level programming language, Python also has a main advantage of readability. In this study, as the focus is sharpened to the calculations of equation for distributed formation control, python will be a more suitable language to manage the equations compared to C++.

3.2.2 TurtleBot3

The model of choice for swarm robots used in MAS is turtlebot3 burger model manufactured by ROBOTIS based in Korea. Turtlebot3 is a complete packaged hardware composed of the frame, two DC DYNAMIXEL servo motors, 2D LiDAR, on-board augmented Arduino mega controller board and a Raspberry Pi. In Figure 3.2, a TurtleBot3 Burger is shown with dense components.

Figure 3.2: Isometric view of a TurtleBot3 Burger

3.2.2.1 Hardware specifications

The robot has the following hardware specification:

Table 3.1: Hardware specifications of Turtlebot3

Item	Turtlebot3
Maximum translational velocity	0.22 m/s
Maximum rotational velocity	2.84 rad/s
Size (Length x Width x Height)	138 mm x 178 mm x 192 mm
Actuator	Dynamixel XL430-W250
Laser Distance Sensor	360 Laser Distance Sensor LDS-01
Inertial measurement unit (IMU)	Gyroscope 3 Axes Accelerometer 3 Axes Magnetometer 3 Axes

Since this robot comes with configured drivers, some technical problems which go beyond the scope of this study such as networking and low-level communications between circuit boards could be bypassed.

The packages provided by ROBOTIS already integrate the hardware connection between the on-board computer (Raspberry Pi) and the controller board (Arduino) to ROS. The complete solution makes this robot easy for evaluating control methods used and eliminating the hassle of troubleshooting buggy observations caused by unreliable integrations.

Hence, the movement of turtlebot3 is remotely controlled by publishing specified velocity commands to the robot itself, as this study's sample applications do not require on-board computation. The on-board computer is responsible only for communication and powering the two wheels on the differential drive servo motors.

This model of the robot is also chosen to be the simplest representation of a differential drive robot, without the need to transpose or alleviate the error caused by different forms of wheeled or legged drives.

In terms of control, since Turtlebot3 is controlled under ROS, most numerical data of the robot can be extracted and viewed under a provided Graphic User Interface (GUI).

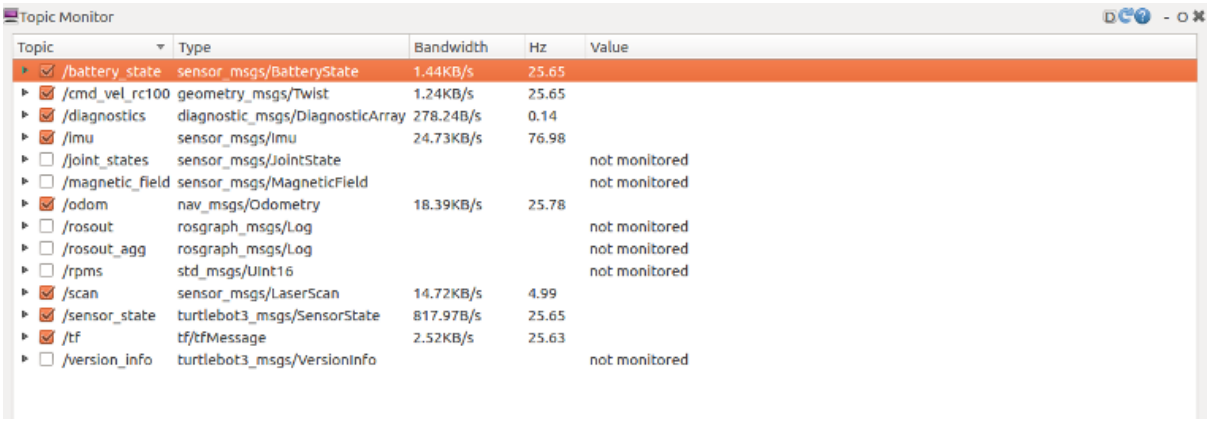

Figure 3.3: Topic in ROS Communication Network

"Turtlebot3" will be regarded as "robot" or "TB3" hereafter.

3.2.2.2 Communication Setup

To connect different TB3s to the ROS network, the configuration of each TB3 was modified. By modifying the file ".bashrc", which will be executed when the terminal in Linux OS is used.

```
export ROS_MASTER_URI = http://192.168.1.180:11311
export ROS_IP = 192.168.1.181
```

The above commands were used to connect a TB3 to the master PC, where the IP of the identical TB3 is 192.168.1.181, and the IP of master PC is 192.168.1.180.

3.2.2.3 Software Setup

With open-source TB3 calibration packages, the sensors, motor, lidar can be calibrated automatically and the robot will be ready for motion control. In addition, some important variables of TB3 will be published to the ROS network, including ODOM and battery state.

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
multi_robot_name:="tb3_1"
```

With the ROS network connecting the master and the robots, to exchange information between robots and computers, publisher and subscriber are used in ROS system.

Table 3.2: Common terms and descriptions in ROS

Terms	Description
Publisher	Publisher is a ROS data exchange term which means continually broadcast a message to topic in the ROS network
Subscriber	Subscriber is a ROS data exchange term which means continually receive a message to topic in the ROS network
Message	Message is a ROS data type used when subscribing or publishing to a topic.
Topics	Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.

3.2.3 OptiTrack Motion Capture System

3.2.3.1 OptiTrack Hardware Setup

The OptiTrack system is an indoor motion-capturing system which captures marked objects movement. In the study, the marked objects are the arbitrary spheres mounted on the robots in a unique pattern. The system is suitable to be the primary feedback device capturing the position data of the robots in the laboratory. In Figure 3.4, the layout of OptiTrack system is shown.

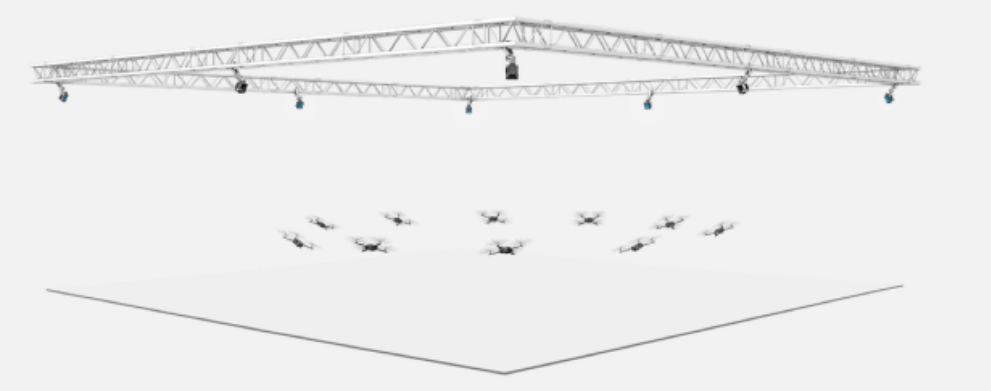

Figure 3.4: Setup of OptiTrack motion capture system indoor

Figure 3.5: A Prime 13 motion camera (Left) and the setup in laboratory (Right)

OptiTrack system comprises three hardware components, one important component is the motion capture camera. In this study the Prime 13 camera was used, and an array of 10 cameras were placed in the laboratory to track the testing area in the laboratory. The camera and setup in laboratory are shown in Figure 3.5.

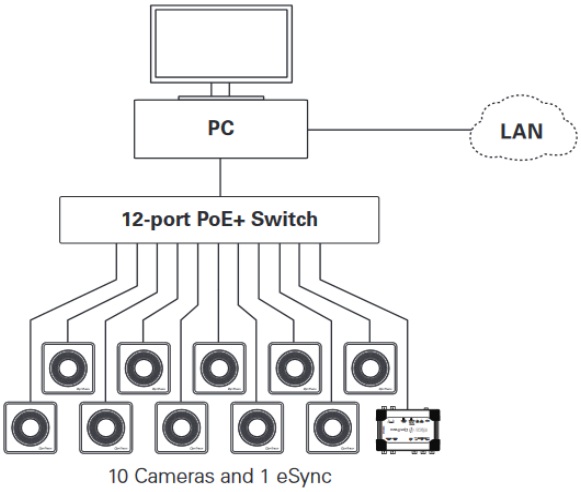

Figure 3.6: Connection of Optitrack Motion Capture System

The motion cameras receive the data and send to the host PC at their maximum capture rate of 250fps, through an uplink PoE device connected to the host PC via LAN connection. The PoE device also serves as the power source of the cameras.

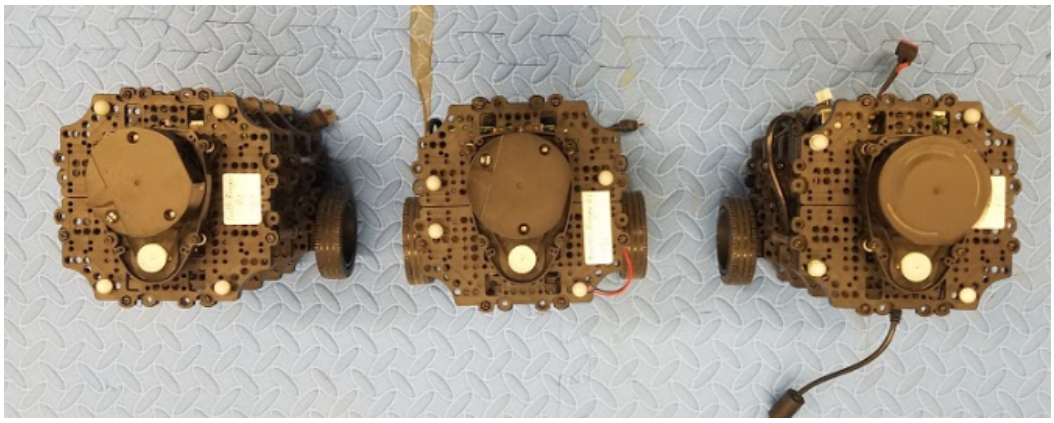

Figure 3.7: Spherical reflective marker mounted on the robot in different pattern

To identify different agents in the system, the spherical reflective markers were mounted in a unique pattern. The motion capture software is able to classify the motion of the identical robot and deliver the position information to the master computer.

3.2.3.2 OptiTrack Software Setup

Motive 1.8 software was installed on the host PC to process the data from the cameras. The marker orbs on the robots are reflective to the camera's LED and multiple markers create different degrees of distorted patterns, the software differentiates the difference between the

images and calculates the distance of the object (multiple markers placed constantly apart) with respect to different cameras. Despite physical constraints blocking some markers to be read by some cameras, the redundancy required to produce an accurate position is usually >3 markers with >5 cameras' POV. The rigid body function is used to define the robot in the indoor space, and the fluctuation of a still object is usually less than 2mm.

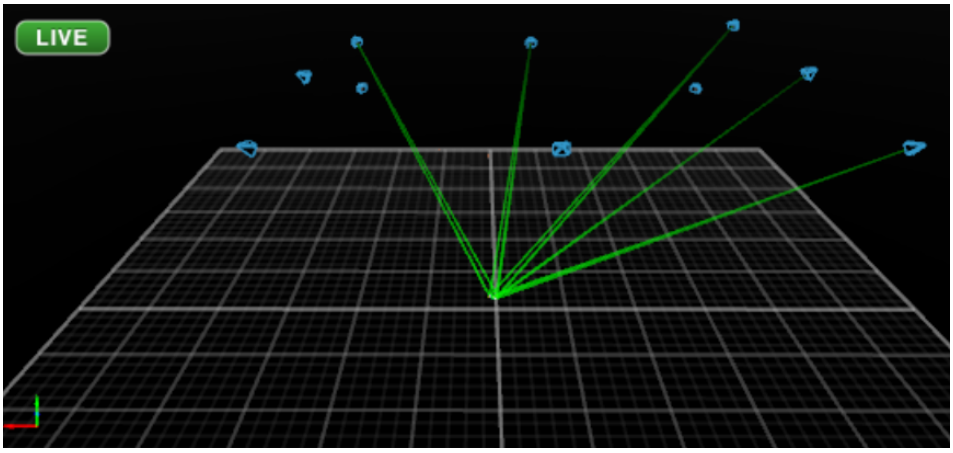

Figure 3.8: OptiTrack Motion Capture Software (Motive 1.8) information

Despite the use of this software, the analysis of the positional data was done with ROS. This requires either running ROS natively on Windows OS for compatibility reasons or using the data streaming function and transfer the data on the client PC running the OS of choice.

The latter approach was chosen because of the limited support of robots to the first beta versions run on Windows OS. Motive 1.8 supports streaming via OptiTrack's own protocol NatNet, which is then decoded with an open-source third-party package native to ROS Kinetic Version running on Linux OS on the client PC. It is done through the router connection in the laboratory.

3.2.3.3 OptiTrack Data Streaming in ROS network

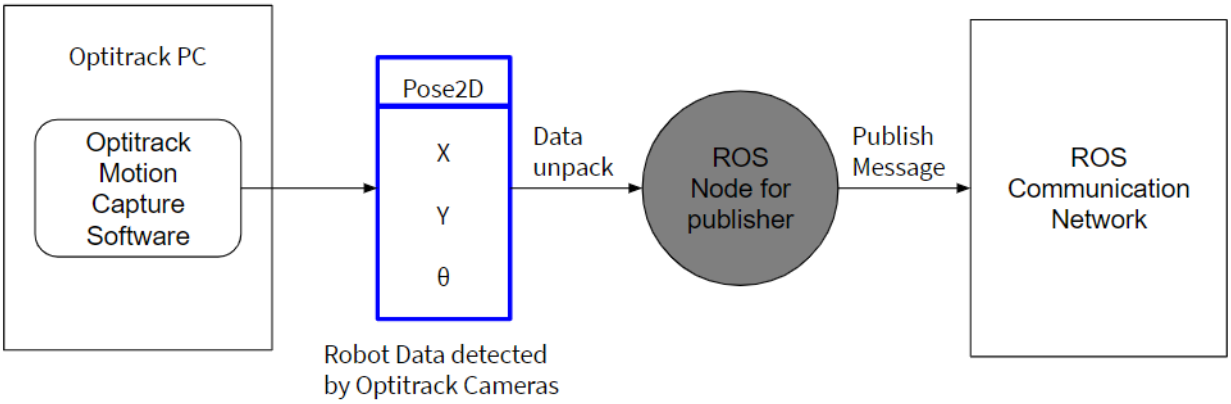

Figure 3.9: Flow Chart of Data Streaming from OptiTrack software to ROS network

In Figure 3.9, the flow diagram of data stream from OptiTrack software to ROS network is shown. With user input configuration, the data number defined in OptiTrack software, data needed front the OptiTrack software and the desired published topic name in ROS network are defined. In this study, the data type [Pose2D] from the class library [geometry_msgs] was used, the data was transferred with data packages, which contains the data $[x, y, \theta]$. The data packages would then be unpacked and passed into ROS node for publisher, which would publish the related data to ROS communication network with user-defined topic name.

3.2.4 Setup Summary

In Figure 3.7, the reflective marker orbs were placed on turtlebot3 as OptiTrack needs to observe a set pattern to trace the object. Motive distinguished different robot by observing the difference in the marker arrangements. Motive then streamed the position and orientation of the robots to the host PC via laboratory connection.

The robots were connected to the host PC via a ROS master-slave connection. Each robot was configured to search for the host PC IP address in their file. Shared ROS data types were then available for use on multiple machines.

Specifically, to control turtlebot3, ROS was set to the Kinetic distribution and run natively on Linux Ubuntu 16.04 version. The foundation package is referenced from the package contained in the manual for turtlebot3. Stream data was then published from OptiTrack to the host PC via

network connection to the designated topic, and the result was viewed via visualized GUI tools provided in ROS such as RViz and rqt.

After the above items were running, codes presenting different tasks of controller would then be executed in ROS launch files. The running codes were scripted in Python.

3.3 TB3 Control Algorithms

To achieve a controllable architecture for a two-wheeled UGV dynamics, the basic dynamics for a two-wheeled robot will be mentioned. Subsequently, a control scheme base of the dynamics will be derived. Lastly, the method for path planning will be introduced.

There are many types of UGVs, including differential drive robots and steering vehicles. TB3 Burger is a common type of differential drive robots. Therefore, the dynamics introduced in this section will focus only on differential drive robots.

3.3.1 Kinematics of Differential Drive Robots

Since TB3 Burger is a type of differential drive robots, the pose of TB3 Burger can be represented by location and orientation which are determined by frame state $\xi = [x, y, \theta]^T$ where x, y are the x,y-coordinates and θ is the orientation of TB3 Burger.

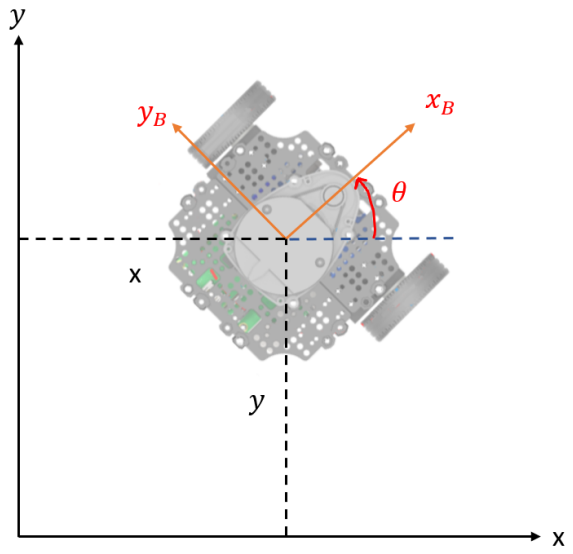

Figure 3.10: Demonstration of TB3 Burger with $\xi = [x\ y\ \theta\ ]^T$

In Figure 3.10, a TB3 Burger with frame state $\xi = [x, y, \theta]^T$ is shown. x, y are the x,y-coordinates of global Cartesian Frame where x_B, y_B are the x,y-coordinates of local Cartesian Frame of the robot. The orthogonal rotation matrix $R(\theta(t))$ between two frames could be given by:

$$\begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = R\big(\theta(t)\big) \begin{bmatrix} x_B(t) \\ y_B(t) \\ \theta_B(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_B(t) \\ y_B(t) \\ \theta_B(t) \end{bmatrix} \tag{3.1}$$

The kinematics equations of a differential driven robot are given by:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos\big(\theta(t)\big) & 0 \\ \sin\big(\theta(t)\big) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.2}$$

where $v(t)$ is the linear velocity and $\omega(t)$ is the angular velocity of the differential driven robot.

Since a differential drive is driven by two independent wheels, the linear velocity $v(t)$ and angular velocity $\omega(t)$ can be related with the parameters of the wheels. A standard differential driven robot should have its wheels of the same radius, such that $r_L = r_R = r$. The inverse kinematics of differential drive robot is given by:

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ -\frac{r}{L_w} & \frac{r}{L_w} \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \tag{3.3}$$

where L_w is the distance between the wheels. Correspondingly, the forward kinematics of differential drive robot could be given by:

$$\begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & -\frac{L_w}{2r} \\ \frac{1}{r} & \frac{L_w}{2r} \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.4}$$

Combining the above, the following will be obtained:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos(\theta(t)) & \frac{r}{2}\cos(\theta(t)) \\ \frac{r}{2}\sin(\theta(t)) & \frac{r}{2}\sin(\theta(t)) \\ -\frac{r}{L_w} & \frac{r}{L_w} \end{bmatrix} \begin{bmatrix} \omega_L(t) \\ \omega_R(t) \end{bmatrix} \tag{3.5}$$

which relates the directional velocities to the angular velocities of the wheels and the robot orientation only.

3.3.2 Position Control of Differential Driven Robots

With the kinematics equations of differential drive robot in section 3.3.1, the position control of the robot will be introduced in this section.

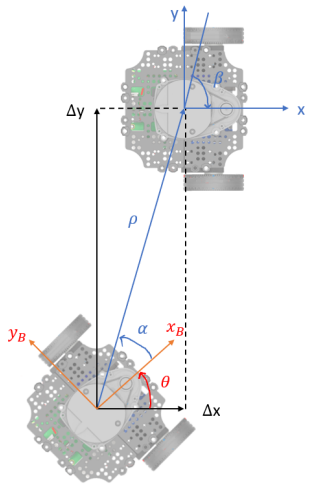

Figure 3.11: Demonstration of TB3 Burger moving from current pose to goal pose

Using the notation shown in Figure 3.11, x_B, y_B are the local coordinates of TB3 Burger at current pose and x, y are the global coordinates. The variables are defined by:

$$\rho(t) = \sqrt{\Delta_x(t)^2 + \Delta_y(t)^2} \tag{3.6}$$

$$\alpha(t) = \tan^{-1}\left(\frac{\Delta_y(t)}{\Delta_x(t)}\right) - \theta(t) \tag{3.7}$$

$$\beta(t) = -\theta(t) - \alpha(t) \tag{3.8}$$

With equations (3.5), (3.6) and (3.7), the coordinate parameters can then be related to the velocity profiles:

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} k_\rho \rho(t) \\ k_\alpha \alpha(t) + k_\beta \beta(t) \end{bmatrix} \tag{3.9}$$

which are guaranteed to be stable as long as the parameters $k_\rho, k_\alpha, k_\beta$ satisfy the following conditions:

$$k_\rho > 0 \tag{3.10}$$

$$k_\beta < 0 \tag{3.11}$$

$$k_\alpha - k_\rho > 0 \tag{3.12}$$

The body velocities $v(t), \omega(t)$ obtained by equation (3.9) are then be converted to wheel velocities by forward kinematics of differential drive robot mentioned in equation (3.4).

The velocity profiles of the wheels are followed by using independent Proportional-Integration-Derivative (PID) controllers in velocity loop and current loop. The PID controller is given by:

$$u(t) = K_P e(t) + K_D \frac{de(t)}{dt} + K_I \int_0^t e(t)\, dt \tag{3.13}$$

where $u(t)$ is output of the controller and $e(t)$ is the error input of the controlled parameter.

With equation (3.1), along with some rearrangements:

$$\dot{\vec{r}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} \tag{3.14}$$

And for the robot pose $\vec{r}(t)$:

$$\vec{r}(t) = \begin{bmatrix} x(t) \\ y(t) \\ \theta(t) \end{bmatrix} = \int_0^t \dot{\vec{r}}(t)\ dt \tag{3.15}$$

The equations (3.14) and (3.15) together give the state-space model for single robot dynamics control. However, the position $\vec{r}(t)$ can be obtained through the OptiTrack system directly. Therefore, equation (3.15) will not be implemented in experiment. Figure 3.12 below shows a control block diagram for the robot dynamics control:

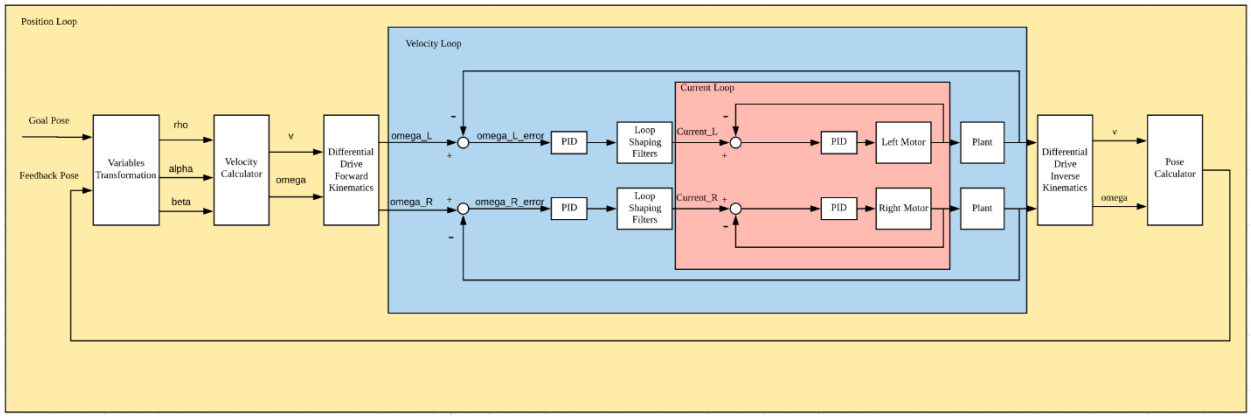

Figure 3.12: Control block diagram of single robot dynamics

Position control of the differential drive robot is the prerequisite of path tracking. In this study, path tracking would be conducted by using position control to track consecutive points.

From equation (3.9), since the linear and angular velocities of the robot are calculated from the Euclidean distance and angle between the goal pose and the current pose, any external disturbances or unpredicted errors would hinder the estimation of velocities of the robot. Hence, the position of robot and motion time could not be estimated accurately. The navigation behaviors of the robot are not fully predictable. Therefore, path planning is introduced in next section for obtaining more predictable motion performances of the robot.

3.3.3 Path Planning

Path planning algorithm is the motion planning of the robot from point to point such that a sequence of points can be calculated in a time-controlled manner. To find the optimal path, different algorithm will implement diverse cost function, and ensure the result path having the minimum cost. Therefore, with the calculated list of points and the use of position control method, motion between points can be performed.

3.3.3.1 Quintic polynomials planning

In Quintic Polynomials Algorithm, with the use of derived fifth order polynomials equation, position, velocity, acceleration and jerk of robot at a certain time.

The equation for 1-D motion of a robot can be derived and represented by:

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{3.16}$$

where $s(t)$ is the position of the motion with respected to time. With the boundary conditions of initial and final positions, velocities and acceleration, the coefficients in equation (3.16) could be solved correspondingly. The representation of the coefficients $a_0 - a_5$ can be calculated by solving six simultaneous equation satisfying boundary conditions:

Table 3.3: Simultaneous Equation of Quintic Polynomials Algorithm

Coefficient	Simultaneous Equation
a_0	$s(t = 0) = initial\ position$
a_1	$\dot{s}(t = 0) = initial\ velocity$
a_2	$\ddot{s}(t = 0) = initial\ acceletation$
a_3	$s(t = T) = position\ at\ t = T$
a_4	$\dot{s}(t = 0) = velocity\ at\ t = T$
a_5	$\ddot{s}(t = 0) = acceletation\ at\ t = T$

To perform 2-D motions of a robot, the kinematics equations of the robot could be given by:

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{3.17}$$

$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \tag{3.18}$$

$$\theta(t) = \tan^{-1}\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right) \tag{3.19}$$

The cost function C is defined as the time integral of the square of jerk when the robot is moving between the two positions within a time interval T, where T is the total time of the motion:

$$C = \frac{1}{2}\int_0^T \left(\frac{d^3x}{dt^3}\right)^2 + \left(\frac{d^3y}{dt^3}\right)^2 \, dt \tag{3.20}$$

In the equation (3.20), x and y indicate the current robot position coordinates, and the local path is generated by minimizing the cost function.

The calculated path can be verified by calculating the trajectory error e between the robot's current position $(x_s, y_{s)}$ and calculated desired position (x, y, θ) at a certain time interval.

$$e = (x_s - x)\, cos\, \theta + (y_s - y)\, sin\, \theta \tag{3.21}$$

If the magnitude of $|e| > desired\ error\ value$, the local path will be re-generated with the latest measured data. If $|e| < desired\ error\ value$, the path calculation will continue without restraining the trajectory error.

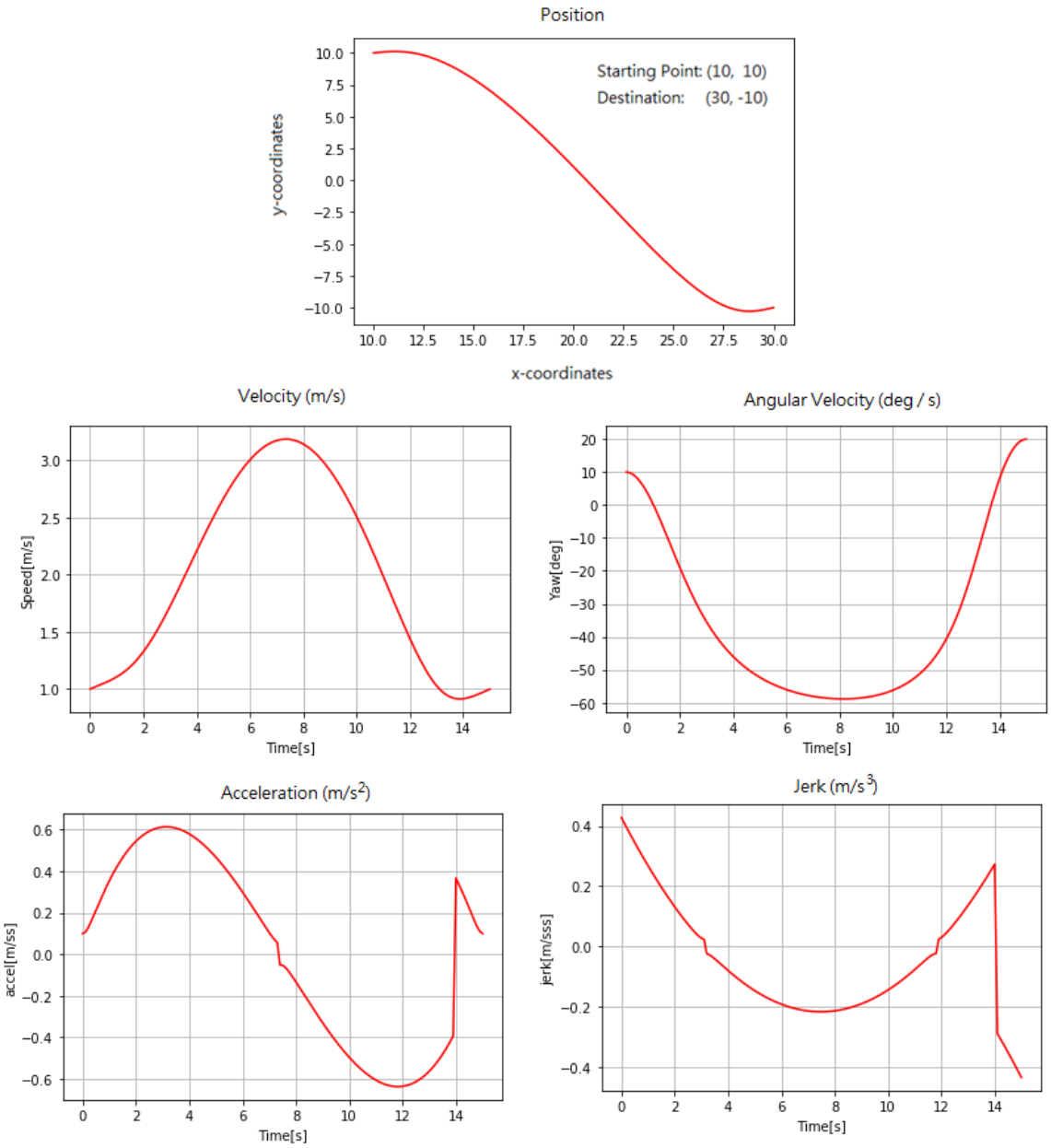

Figure 3.13: Simulation of Quintic Polynomial Path planning

Figure 3.13 shows the simulation path planning results with python. In the figure, the position, velocity and turning angle results of Quintic Polynomial Path Planning are shown, with starting coordinate at $(10, 10)$ and desired destination at $(30, -10)$. In the calculation of path, $x(t)$ and $y(t)$ were minimized in the time integral of the square jerk by the derived fifth order polynomials. In addition, the first and second derivatives in the polynomial are continuous, therefore, the velocity and curvature are also continuous.

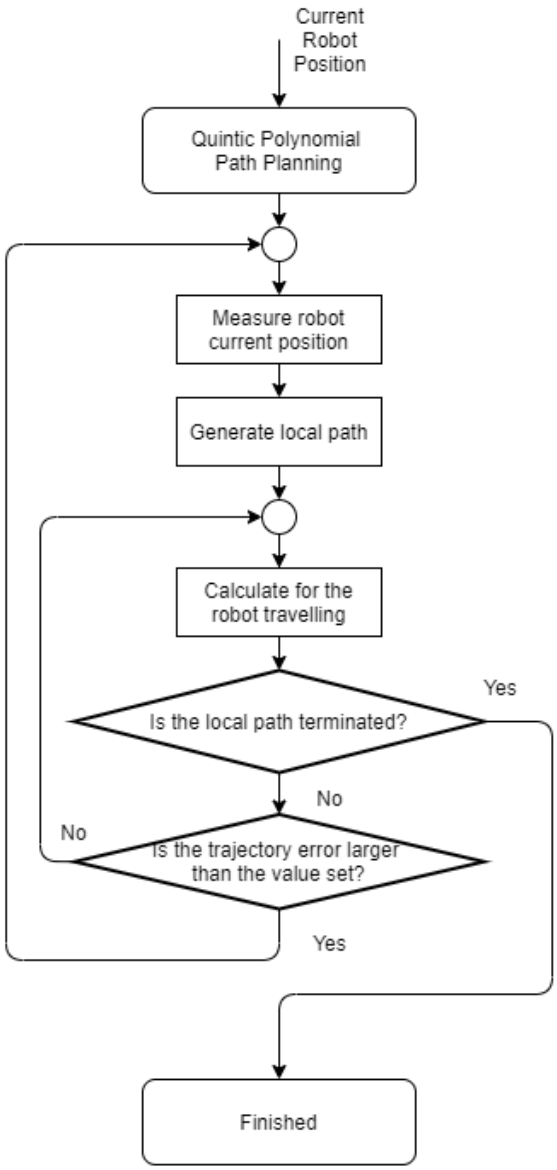

Figure 3.14: Flow chart of Quintic Polynomial Path Planning

In Figure 3.14, the flowchart of Quintic Polynomial Path Planning is shown. The function will operate with the feedback position of the robot and the desired destination. After calculation of each step, the algorithm will compare if the current calculating point is the goal, if yes, the algorithm will return the calculated path, if no, the function will continue the calculation process.

3.3.3.2 A* algorithm

A* algorithm is a path searching algorithm by combining Dijkstra's algorithm and Greedy Best-First-Search algorithm. A* algorithm can return the shortest path with a shorter processing time, and able to perform collision avoidance with obstacles' coordinates. Therefore, A* algorithm was studied and considered as an alternative path planning method in the study.

In the path planning algorithm, heuristic distance and actual distance are used to calculate the path.

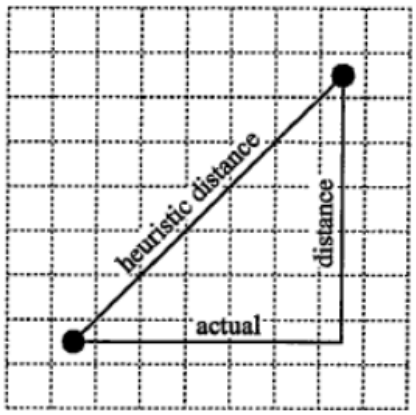

Figure 3.15: Heuristic distance and actual distance in graph calculation

In Figure 3.15, the calculating method of heuristic distance and actual distance between nodes are shown. The distance definition is used in cost function to define the optimal path in different algorithm.

Dijkstra's algorithm is an algorithm used to calculate the shortest path between nodes in a graph. Dijkstra's Algorithm works by visiting vertices in the graph starting with the robot's starting point, current position is treated as the starting point in the study. Then, the algorithm will repeatedly examine the closest not-yet-examined vertex, adding its vertices to the set of vertices to be examined. It expands outwards from the starting point until it reaches the goal, and the list of points will be the path calculated. Dijkstra's Algorithm is guaranteed to find a shortest path from the starting point to the goal, while it is using the actual distance as distance reference.

The cost function for Dijkstra's algorithm is calculated with actual distance of the node from the goal, and can be determined as:

$$C(n) = g(n) \tag{3.22}$$

Where n is the visiting node, $g(n)$ is the actual distance of the node from the goal.

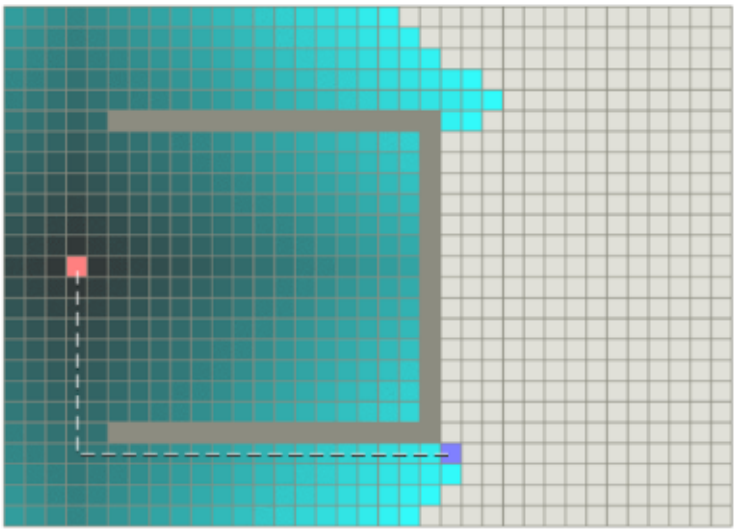

Figure 3.16: Searching Simulation of Dijkstra's algorithm

In Figure 3.16, the searching simulation of Dijkstra's algorithm is shown, where the red grid is the starting point, the blue grid is the goal point, the dotted line is the optimal path, the colorized grids are the visited node. The algorithm can ensure the optimal path is the shortest path, but the processing time is longer compared to Greedy Best-First-Search algorithm and A* Algorithm.

Greedy Best-First-Search algorithm works in a similar way in the iteration of point search will continue until destination is reached. However, Greedy Best-First-Search employs an estimated distance, which is called heuristic distance to estimate how far from a point is away from the goal. Instead of selecting the vertex closest to the starting point, it selects the vertex closest to the goal. Greedy Best-First-Search is not guaranteed to find a shortest path. However, the time it needs to return the path is quicker than Dijkstra's Algorithm because it uses the heuristic function to guide its way towards the goal very quickly.

The cost function for Greedy Best-First-Search algorithm is calculated with heuristic distance of the node from the goal, and can be determined as:

$$C(n) = h(n) \tag{3.23}$$

Where n is the visiting node, $h(n)$ is the heuristic distance of the node from goal position.

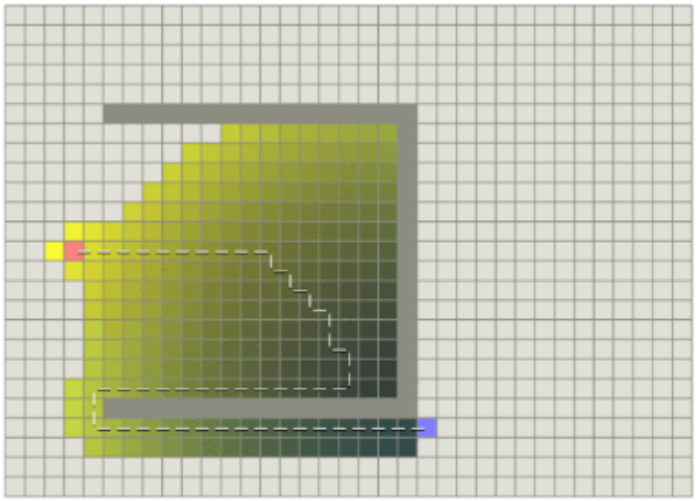

Figure 3.17: Searching Simulation of Greedy Best-First-Search algorithm

In Figure 3.17, the searching simulation of Greedy Best-First-Search algorithm is shown, where the red grid is the starting point, the blue grid is the goal point, the dotted line is the optimal path, the colorized grids are the visited node. The algorithm cannot ensure the optimal path is the shortest path, but the processing time is short compared to Dijkstra's algorithm.

As a combination of Dijkstra's algorithm and Greedy Best-First-Search algorithm, A* has the performance as Dijkstra's Algorithm that it can ensure the optimal path is the shortest path, and have a shorter processing time compared to Dijkstra's algorithm. The cost function for A* algorithm is calculated with both actual distance and heuristic distance of the node from the goal, and is determined as:

$$C(n) = g(n) + h(n) \tag{3.24}$$

Where n is the visiting node, $g(n)$ is the actual distance of the node from the goal, $h(n)$ is the heuristic distance of the node from goal position.

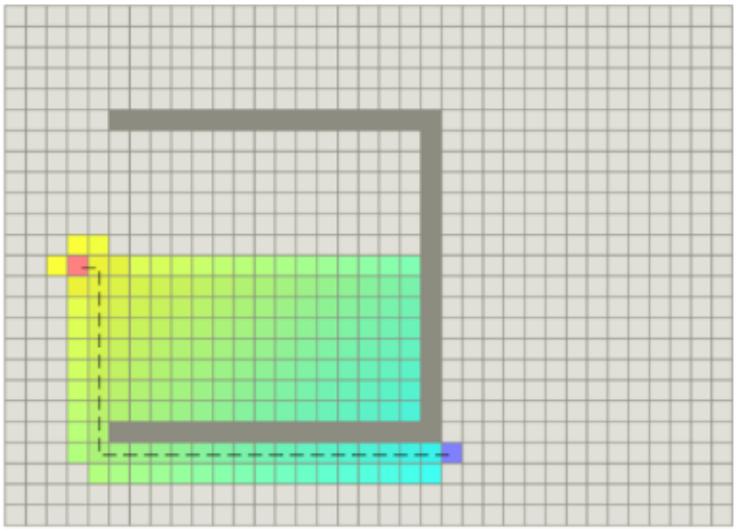

Figure 3.18: Searching Simulation of A* algorithm

In Figure 3.18, the searching simulation of A* algorithm is shown, where the red grid is the starting point, the blue grid is the goal point, the dotted line is the optimal path, the colorized grids are the visited node. The algorithm can ensure the optimal path is the shortest path, and the processing time is short compared to Dijkstra's algorithm.

3.3.3.3 Potential Field algorithm

Potential Field Path Planning algorithm is an algorithm to calculate the robot path to reach the destination and avoid the obstacles in its path. The idea of potential field is similar to a charged particle, it will be attracted or repulsed by the magnetic field. In path planning calculation, the robot path can be created by creating a potential field that will attract the robot to arrive the destination.

In the potential field algorithm, potential will be calculated for each point in the space graph, obstacles will induce repulsive potential and available points will induce attractive potential. The equation is determined as:

$$U_{rep}(n) = \frac{1}{2}\eta\left(\frac{1}{h(n)} - \frac{1}{R}\right)^2 \tag{3.25}$$

$$U_{att}(n) = h(n) \tag{3.26}$$

Where η is a constant, R is the radius of obstacles, $h(n)$ is the heuristic distance between current node and goal position.

The cost function C is considering the sum of attractive and repulsive potential to calculate the optimal path and is determined as:

$$C(n) = U_{rep}(n) + U_{att}(n) \tag{3.27}$$

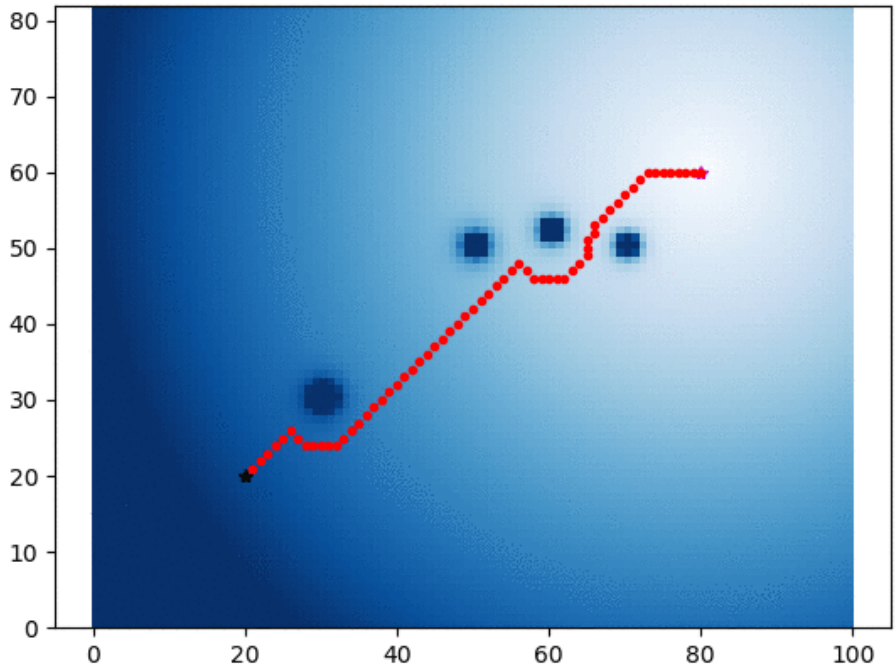

Figure 3.19: Example of Potential Field Algorithm

In Figure 3.19, the simulation result of potential field algorithm is shown. The starting point of the path was at (20, 20) and the goal position was at (80, 60). In the calculated potential field map, the obstacles will have repulsive potential (dark blue color) and the point near the robot will have attractive potential (white color). The path of the robot was calculated based on the potential field. Therefore, obstacle avoidance could be obtained by the algorithm as the area near the obstacles would not be considered as a part of the path.

3.3.4 Simulation Tests

Simulation tests by Simulink will be conducted to validate the position control algorithms of single differential drive robot. The influences of the values of gains related to velocities in equation (3.8) will also be investigated by the simulation tests with navigating from a specific point to another.

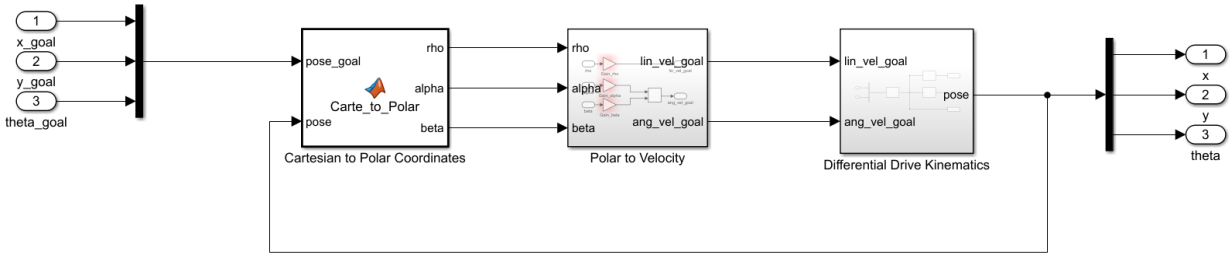

Figure 3.20: Block diagram of position control in Simulink

In Figure 3.20, the block diagram of position control in Simulink shown is equivalent to the control block diagram shown in Figure 3.10. The inputs of the simulation would be the initial and goal poses which are consisted of x, y-coordinates and θ-orientation respectively. The outputs of the simulation would be the pose of the robot.

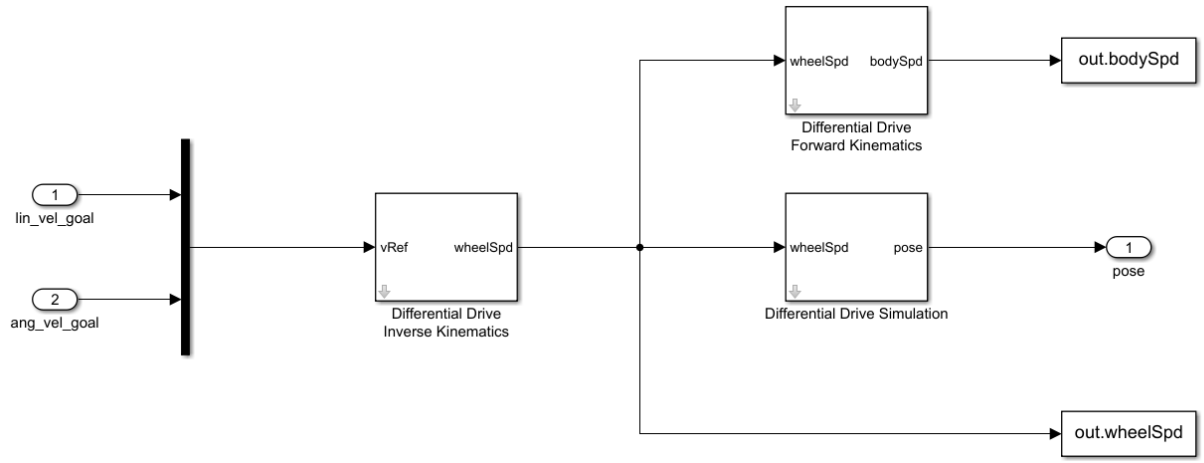

Figure 3.21: Block diagram of differential drive kinematics in Simulink

In Figure 3.21, the details of block "Differential Drive Kinematics" are shown in Figure 3.20. The block is consisted of forward kinematics in equation (3.4), inverse kinematics in equation (3.3) and the simulation of a differential drive robot.

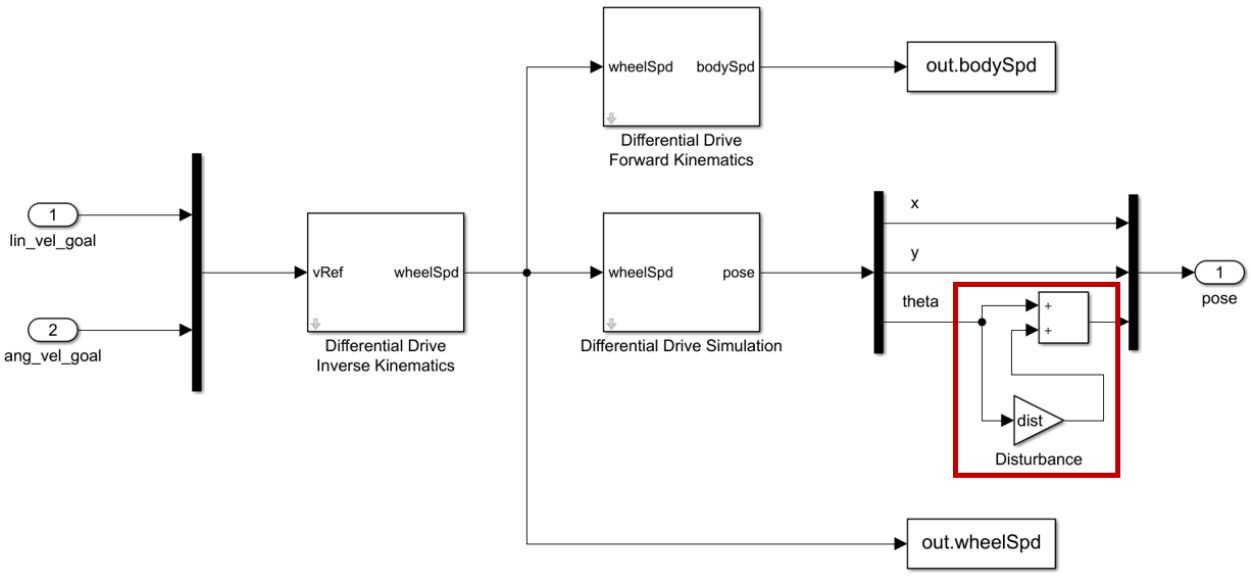

Figure 3.22: Block diagram of differential drive kinematics with disturbance in Simulink

Apart from the gains related to velocities, the influences of disturbance of performance of position control. In this study, the disturbance was simulated by adding constant gain to the resultant θ-orientation which is shown in Figure 3.22 with red rectangle bounded. Since linear and angular velocities of the robot are dependent on the values of $\theta(t)$. The disturbance on θ-orientation would also alter the resultant motion of the robot.

3.3.5 Navigation Tests

Navigation tests were conducted to validate the algorithms of path planning and path tracking by inputting a goal pose of TB3 Burger. A goal pose was given as the target of a TB3 Burger. Optimal path was then planned and tracked by TB3 Burger. Data of navigation was collected by the ROS data logging system. The advantages of using path planning and path tracking will be discussed based on the experimental results of TB3 Burgers navigation.

3.3.6 Communication of Multiple Robots

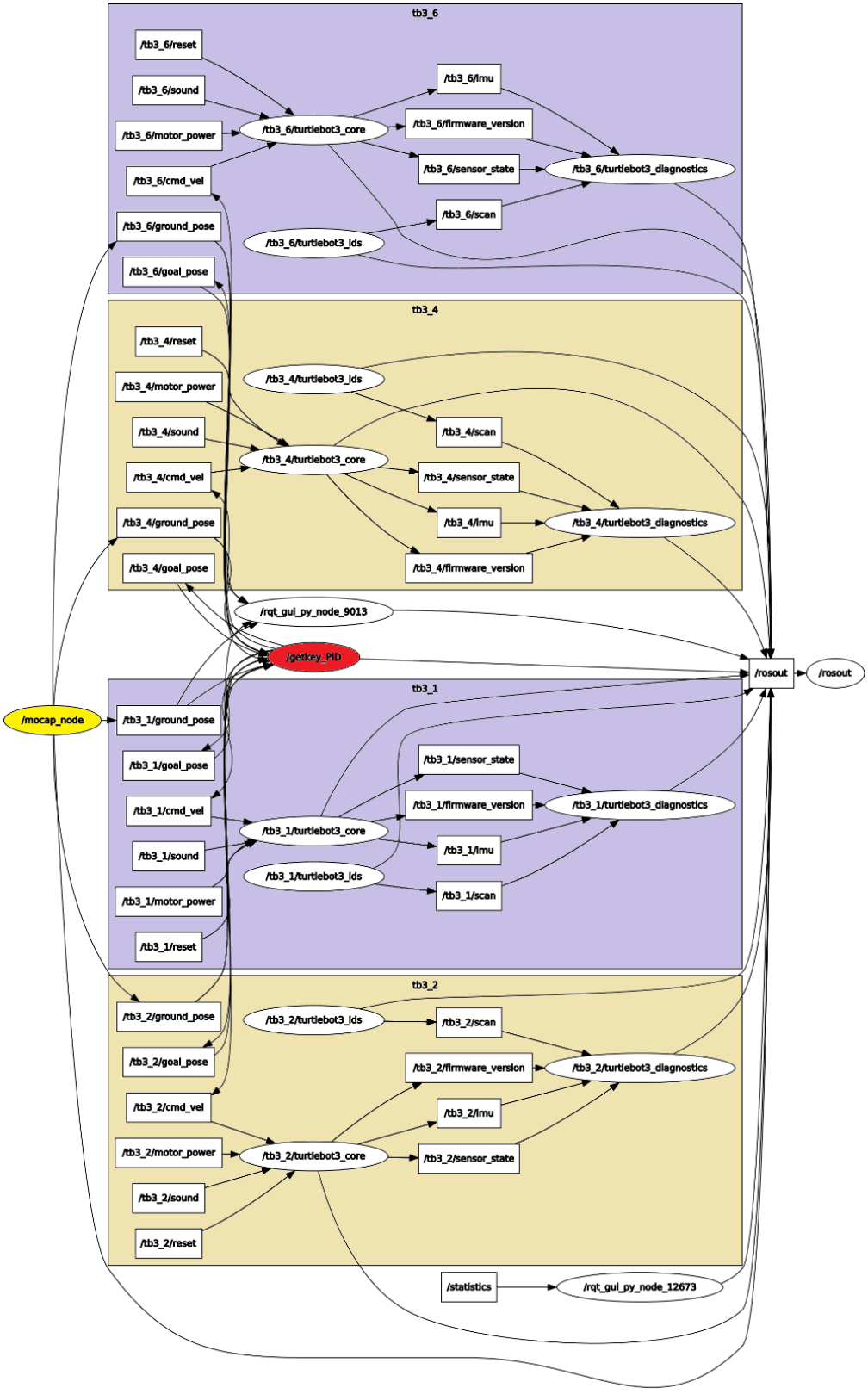

Figure 3.23: Communication diagram between master PC and the robots in ROS system

In Figure 3.23, the node [/getkey_PID] is the node representing master PC, connected by different robots. In addition, the node [/mocap_node] is the node of Optitrack motion capture system, providing the position of the pre-defined robots' neighbors. With the consensus and formation calculation, goal position for different robots are calculated and sent to the connected robot and the formation control could be obtained.

3.4 Design of Formation Control Scheme

As mentioned in Chapter 2.3, virtual leader approach was selected for this study. In the following section, the details of this approach will be explained and the relation between this approach and consensus algorithm will be shown.

For a formation to form, at least one agent in a group has to understand the desired position. This position relies on a formation frame where each agent's desired position relative to the virtual coordinate in the formation frame can be defined. This coordinate usually located at the center of the formation.

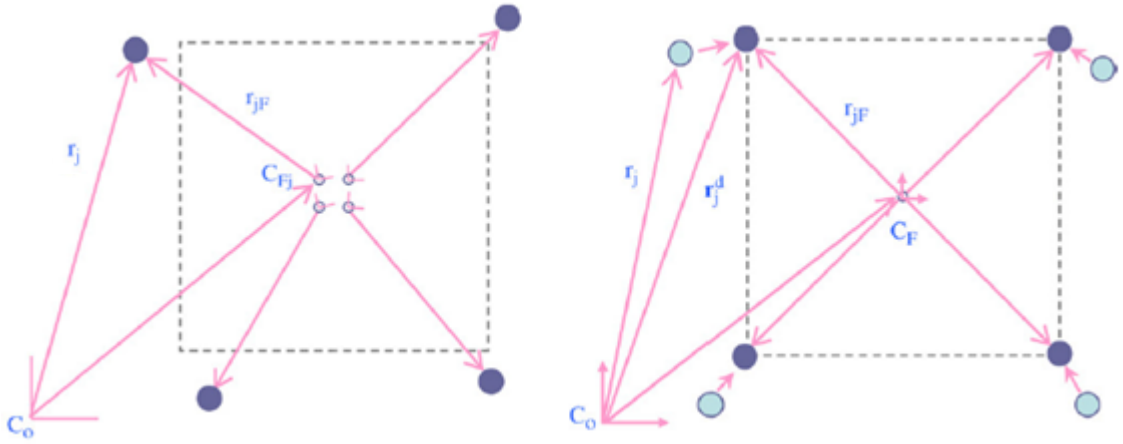

Figure 3.24: A formation composed of four agents with a consistent (right) and inconsistent (left) understanding of a formation frame

Figure 3.24 presents the formation of a square by four agents. The diagram on the left illustrates the moment when these four agents attempt to agree on common ground. At this state, each agent j is at a position:

$$\vec{r}_{jF}(t) = \begin{bmatrix} x_{jF}(t) \\ y_{jF}(t) \end{bmatrix}, \quad j \in \{1, .., n\} \tag{3.28}$$

deviated away from their own understanding of the virtual formation coordinate frame C_{Fj} (F denotes formation) and were at their actual positions:

$$\vec{r}_j(t) = \begin{bmatrix} x_j(t) \\ y_j(t) \end{bmatrix} \tag{3.29}$$

referenced to the global coordinate C_o. The jth agent's understanding of the coordinate C_{Fj} will be denoted as a state:

$$\vec{\xi}_j(t) = \begin{bmatrix} x_{d,j}(t) \\ y_{d,j}(t) \\ \theta_{d,j}(t) \end{bmatrix} \tag{3.30}$$

where x, y denote the x and y position, θ denotes the orientation.

The diagram on the right in Figure 3.24 shows the moment when these four agents eventually reach an agreement about the formation coordinate C_F and each agent j were at their desired positions:

$$\vec{r}_j^{\,d}(t) = \begin{bmatrix} x_j^d(t) \\ y_j^d(t) \end{bmatrix} \tag{3.31}$$

And this desired formation state at $C_F$will be denoted as:

$$\vec{\xi}_d(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \\ \theta_d(t) \end{bmatrix} \tag{3.32}$$

In the transition from the left diagram to the right diagram, it can be seen that this process tries to converge the understanding of individual agents $\vec{\xi}_j$ to the desired formation state $\vec{\xi}_d$. If only one agent i is allowed to know $\vec{\xi}_d$, this agent will have to converge its understanding $\vec{\xi}_i$ towards $\vec{\xi}_d$ ($i \in \{1, .., n\}$), and other agents will have to converge their own understandings $\vec{\xi}_j$ towards $\vec{\xi}_i$ [42].

Intuitively, this operation is similar to that in the consensus algorithm. Since the form of consensus algorithm guarantees the convergence:

$$\vec{\xi}_i(t) - \vec{\xi}_j(t) = 0$$

$$\vec{\xi}_i(t) - \vec{\xi}_d(t) = 0 \tag{3.33}$$

such that:

$$\vec{\xi}_1 = \vec{\xi}_i = \cdots = \vec{\xi}_n = \vec{\xi}_d \tag{3.34}$$

Therefore, the form of consensus algorithm (2.5) can be applied for this formation control problem by treating $x_i(t)$ as $\vec{\xi}_i(t)$. However, this form will not work if a time-varying formation is needed and this form is only valid for formation problems with constant formation center and orientation because the consensus algorithm (2.5) the difference in position will be taken into account [42]. Therefore, after all agents' understanding of the formation state $\vec{\xi}_i(t)$ has been converged to $\vec{\xi}_d(t)$, the formation will hold still and there will be no further action.

To enable time-varying formation, an extended version of consensus law will be needed for the movement (velocity) $\dot{\vec{\xi}}(t)$ of the formation state to be taken into account, both the agents' understandings and the desired state:

$$\dot{\vec{\xi}}_i(t) = \frac{1}{1 + \sum_{j=1}^{n} a_{ij}} \left[\sum_{j=1}^{n} a_{ij} \left[\dot{\vec{\xi}}_j(t) - \varkappa \left(\vec{\xi}_i(t) - \vec{\xi}_j(t) \right) \right] + \dot{\vec{\xi}}_d(t) - \varkappa \left(\vec{\xi}_i(t) - \vec{\xi}_d(t) \right) \right] \tag{3.35}$$

This formula is divided into two parts, the term $\sum_{j=1}^{n} a_{ij} \left[\dot{\vec{\xi}}_j(t) - \varkappa \left(\vec{\xi}_i(t) - \vec{\xi}_j(t) \right) \right]$ comes from the neighbour agents influence and $\dot{\vec{\xi}}_d(t) - \varkappa \left(\vec{\xi}_i(t) - \vec{\xi}_d(t) \right)$ comes from the formation state influence. Since the formation state should only be visible to selected agents, the formation state influence term will not be received by other agents, such that for normal agents [42]:

$$\dot{\vec{\xi}}_i(t) = \frac{1}{\sum_{j=1}^{n} a_{ij}} \sum_{j=1}^{n} a_{ij} \left[\dot{\vec{\xi}}_j(t) - \varkappa \left(\vec{\xi}_i(t) - \vec{\xi}_j(t) \right) \right] \tag{3.36}$$

The convergence to the desired formation state of this extended form of consensus algorithm is also guaranteed [42]. Note that this version of consensus law is averaged by the term $1 + \sum_{j=1}^{n} a_{ij}$ in (3.35) for selected agents and $\sum_{j=1}^{n} a_{ij}$ in (3.36) for normal agents so that the weighting of the consideration of each neighbour and the agent itself is the same, the extra number 1 in equation (3.35) comes from the consideration of the desired formation state.

It is also important to emphasize that since the information topology is not expected to be changed during formation, the adjacency terms $a_{ij}(t)$ in equation (2.5) will not be changed and will remain constant, i.e. $a_{ij}(t) = a_{ij}$.

Each agent j, with their own understanding of the formation frame $\vec{\xi}_j$ and their known relative positions $\vec{r}_{jF}$, can then calculate the desired global position at time t using [42]:

$$\vec{r}_j^{\,d}(t) = \begin{bmatrix} x_{d,j}(t) \\ y_{d,j}(t) \end{bmatrix} + \begin{bmatrix} \cos\big(\theta_d(t)\big) & -\sin\big(\theta_d(t)\big) \\ \sin\big(\theta_d(t)\big) & \cos\big(\theta_d(t)\big) \end{bmatrix} \vec{r}_{jF}(t) \tag{3.37}$$

A formation architecture is also proposed in [42] for distributed formation as follows:

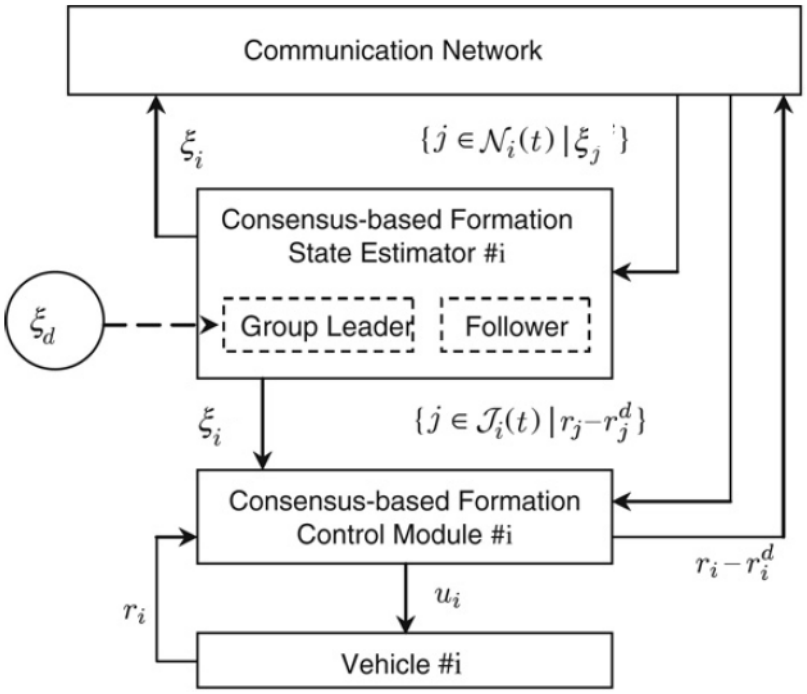

Figure 3.25: Distributed formation control architecture using a virtual leader approach

This architecture (Figure 3.25) puts the three steps mentioned in Chapter 2.3 into more specific tasks which are as follows:

Consensus-based Formation State Estimator

This module aims to negotiate the understanding $\vec{\xi}_i$ of each agent by considering the understanding of neighbouring agents $\vec{\xi}_j$ and the desired formation state $\vec{\xi}_d$ (selected agents only), to drive them towards the desired formation state $\vec{\xi}_d$. This is achieved using equations (3.35) and (3.36).

Consensus-based Formation Control Module

This module gives the required velocity profile u_i for the agent i:

$$\vec{u}_i = \dot{\vec{r}}_i = \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} \tag{3.38}$$

by considering its own understanding of the formation state $\vec{\xi}_i$, the relative distances between each neighboured agent and their own desired positions $\vec{r}_j - \vec{r}_j^d$ and its own ones $\vec{r}_i - \vec{r}_i^d$.

In this study's context, as discussed in section 3.3.2 and 3.3.3, since the location of every agents is available through the OptiTrack system, all the agents' and their neighbors' positional parameters, $\vec{r}_i$ and $\vec{r}_j$, can be easily access through the system without any calculation. An exception is the desired global position $\vec{r}_i^d$ at time t, which is calculated by equation (3.37) using the x and y components in $\vec{\xi}_i$. This $\vec{r}_i^d$ is then passed to the path planning and tracking algorithm to calculate the corresponding linear and angular velocity v_i and ω_i, which in turns gives the velocity profile u_i for the agent i.

Vehicle dynamics

This module will take $\vec{u}_i$ as input to control how the robot moves as mentioned in section 3.3 and return its own position $\vec{r}_i$ through a single integrator transformation:

$$\vec{r}_i = \int_0^t \vec{u}_i \, dt \tag{3.39}$$

But in the context of this study, the positions $\vec{r}_i$ can be obtained from OptiTrack as mentioned above. Therefore, the equation (3.39) will not be necessary.

Note that all the three modules above are implemented within every agent, and information from neighboring agents is received through a communication network, which is a router in this case.

3.5 Summary

An overview of the study setup, the dynamics of the robots used, single robot control scheme and path planning, along with the design of the formation control scheme adopted were given in this chapter. In the next chapter, recent results, limitations discovered and difficulties encountered will be discussed.

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the results of the robot movement in both path tracking method and path planning method presented and discussed at length. In addition, the limitations and difficulties will be discussed.

4.2 Robot Navigation Results

4.2.1 Path Tracking Results of Single Robot

4.2.1.1 Simulation Results

Recall equation (3.9):

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} k_\rho \rho(t) \\ k_\alpha \alpha(t) + k_\beta \beta(t) \end{bmatrix} \tag{3.9}$$

The gains $k_\rho, k_\alpha, k_\beta$ were determined by trial and error with the stability condition in equations (3.9) to (3.11). The values of the gains in Table 4.1 were chosen to be the reference group during the investigation of the influences of the gains in the motion of robot.

Table 4.1: Values of kinematics parameters in MATLAB Simulation

Parameter	Value
$\boldsymbol{k_\rho}$	0.5
$\boldsymbol{k_\alpha}$	8
$\boldsymbol{k_\beta}$	-4

Throughout the simulations, the initial pose and goal pose of path tracking were $(x, y, \theta) = (0.3, 0.4, 30^\circ)$ and $(x, y, \theta) = (0, 0, 0)$ respectively.

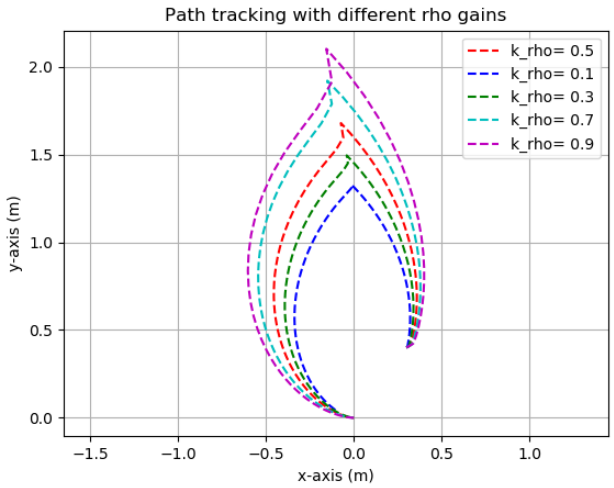

Figure 4.1: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different k_ρ

To investigate the effect of k_ρ on the motion of robot, k_α, k_β were kept at 8, -4 respectively. In Figure 4.1, the motion performances of differential drive robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different values of k_ρ are shown. With larger values of k_ρ, the robot traveled longer distance and the paths would be with smaller curvature. The simulation results agreed with equation (3.8). When k_ρ increased, the robot would possess higher linear velocity gain and higher linear velocity when $\rho(t)$ was identical. Therefore, the increased sensitivity in linear velocity and unchanged sensitivity in angular velocity led to smaller curvature of motions.

Table 4.2: Motion times of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different k_ρ

k_ρ	0.1	0.3	0.5	0.7	0.9
Motion Time	39.9 s	22 s	13.6 s	10.1 s	8 s

However, the smaller values of k_ρ would lead to the increase in motion time since linear velocity decreased. In Table 4.2, the motion times of different values of k_ρ are shown with linear tolerance equal to 0.1 m. The motion time for k_ρ= 0.1 was about 4.99 times of the motion time for k_ρ= 0.9. k_ρ had to be chosen to have balance between motion times and the motion profiles.

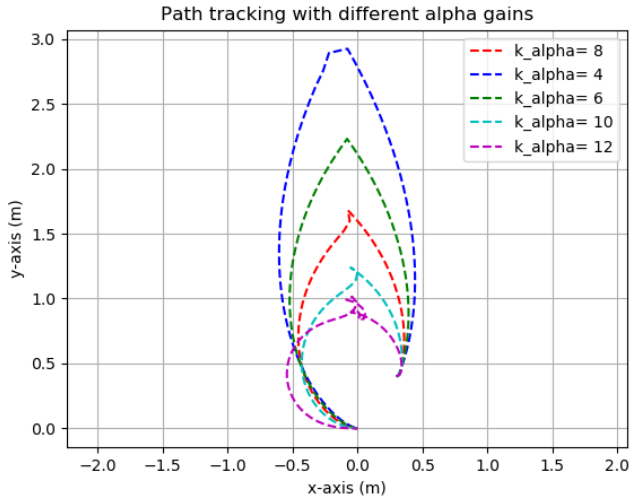

Figure 4.2: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different k_{α}

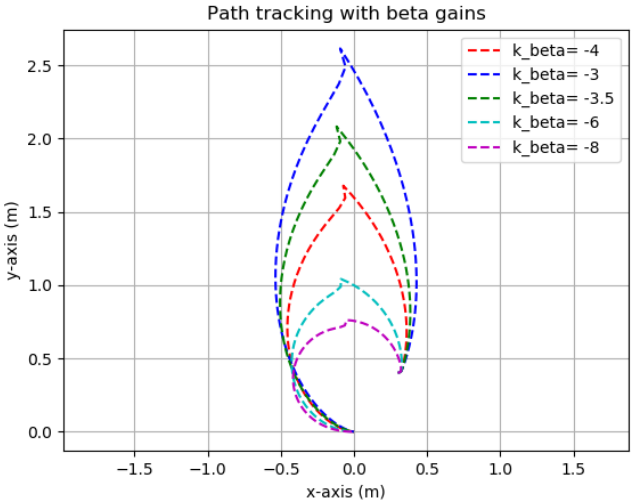

Figure 4.3: Motions of robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different k_{β}

The influences of k_{α}, k_{β} would be considered together since both k_{α}, k_{β} are the gains related to angular velocities.

To investigate the effect of k_{α} on the motion of robot, k_{ρ}, k_{β} were kept at 0.5, -4 respectively. In Figure 4.2, the motion performances of differential drive robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different values of k_{α} are shown.

To investigate the effect of k_{β} on the motion of robot, k_{ρ}, k_{α} were kept at 0.5, 8 respectively. In Figure 4.3, the motion performances of differential drive robot from $(0.3, 0.4, 30^{\circ})$ to $(0, 0, 0)$ with different values of k_{β} are shown.

According to Figures 4.2 and 4.3, the robot traveled shorter distance and the paths would be with larger curvature with higher values of k_α or lower values of k_β. The simulation results agreed with equation (3.8). When k_α increased or k_β decreased, the robot would possess higher angular velocity gain and higher angular velocity when $\alpha(t)$ or $\beta(t)$ are identical. Therefore, the increased sensitivity in angular velocity and unchanged sensitivity in linear velocity led to higher curvature of motions.

However, k_α could not be increased nor k_β could not be decreased infinitely. The higher values of k_α and lower values of k_β would diminish the stability of the differential drive robot. Thus, k_α, k_β had to be selected with the balance of stability and motion profiles.

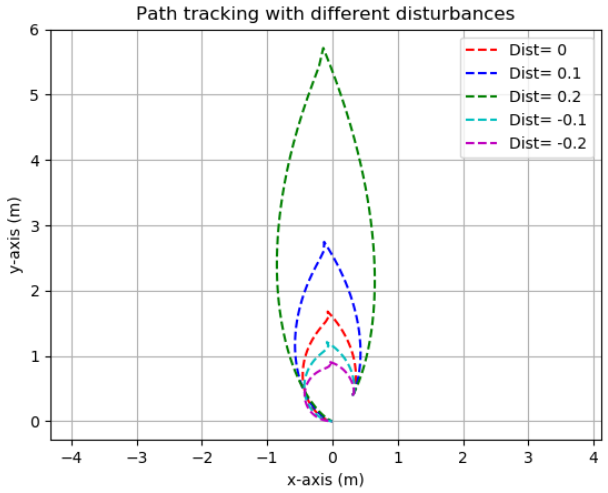

Figure 4.4: Motions of robot from $(0.3, 0.4, 30^\circ)$ to $(0, 0, 0)$ with different disturbances

In Figure 4.4, it can be observed that motions of differential drive robot navigating from $(0.30.4, 30^\circ)$ to $(0, 0, 0)$ were affected with different disturbances in the θ-orientation indicated in Figure 3.17. Any disturbances in feedback loop would greatly affect the motions of the robot. Thus, the motion times were greatly affected.

In these simulations, the motion profiles of path tracking were highly dependent on the values of gains related to velocities and the disturbances which made the motions hard to predict. Therefore, path planning is necessary in order to control the differential drive robot in time-dependent profiles.

4.2.1.2 Experimental Results

With fine tuning of the control parameters based on the findings in simulations, the values of gains related to linear and angular velocities used in experiment are listed in Table 4.3:

Table 4.3: Values of kinematics of TB3 Burger

Parameter	Value
k_ρ	0.8
k_α	5
k_β	-2.5
Distance Tolerance	0.1 m
Angular Tolerance	5°

In Fig 4.5, a dot represents a pose feedback received from OptiTrack. By choosing random initial pose $(x, y, \theta) = (0.3, 0.4, 30^\circ)$ and goal pose $(x, y, \theta) = (0,0,0)$ as shown in Fig 4.4, the TB3 Burger moved as blue curve shown. The TB3 Burger could reach the goal pose by reaching equilibrium at (0,0,0) as introduced in Section 3.3.2. The final pose of TB3 Burger was not exactly equal to the goal pose since Euclidean distance tolerance and angular tolerance were reached under this situation. The shape of the motion in experiment agreed with that in simulations conducted in section 4.2.1.1.

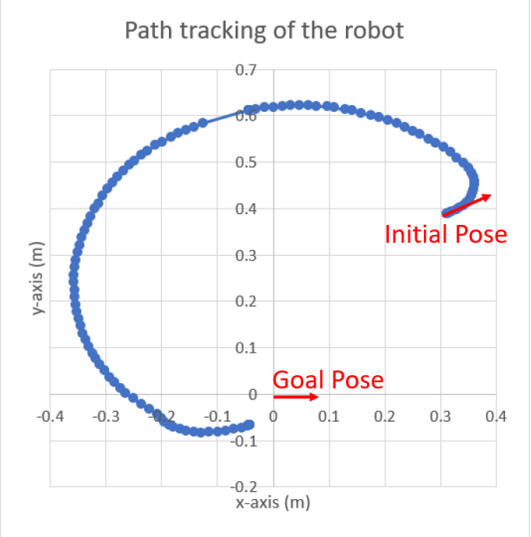

Figure 4.5: Path tracking of TB3 Burger with initial pose and goal pose

4.2.2 Path Planning Results of Single Robot

With the three studied path planning algorithms, different paths were generated and implemented to a single robot. By comparing the desired path and feedback position, the performance of path planning algorithms were determined.

In Figure 4.6, the result of difference between actual motion and desired path using quintic polynomials algorithm is shown. Quintic polynomials algorithm was able to calculate a smooth path, the velocity profile was continuous and the jerk was minimized. Position, velocity and acceleration could be calculated at different time intervals.

The video of the robot motion can be visited through the link: https://youtu.be/ljTQKX0Jqs4.

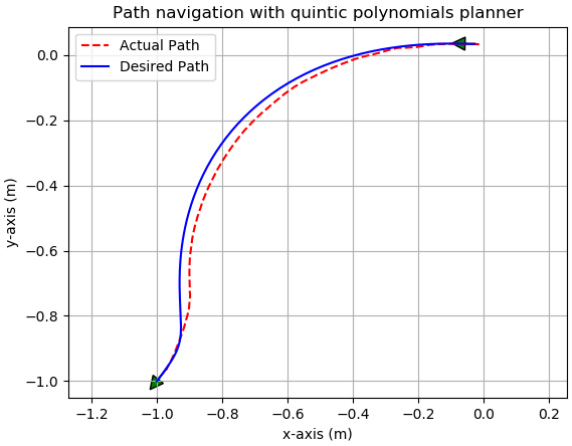

Figure 4.6: Path planning of turtlebot with Quintic Polynomials planner

In Figure 4.7, results of path planning using A* algorithm is shown. A* algorithm ensured the calculated path to be the shortest path, and was able to avoid obstacles with obstacles' coordinates.

The video of the robot motion can be visited through the link: https://youtu.be/HYjw9-yp29w.

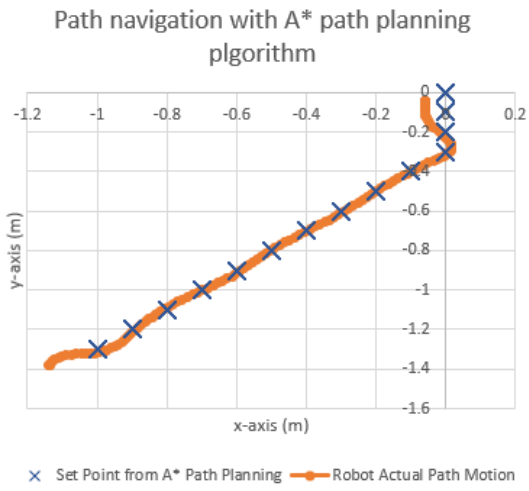

Figure 4.7: Path planning of turtlebot with A* Path Planning Algorithm

In Figure 4.8, the actual motion path calculated by potential field algorithm is shown. The blue points are the motion track of robot 1, while the other robots (robot 2, 3, 4) were static. As a result, the navigating robot could avoid the obstacles with reference to the radius of other robots and reached the goal.

The video of the robot motion can be visited through the link: https://youtu.be/kVsulkE3Eyo.

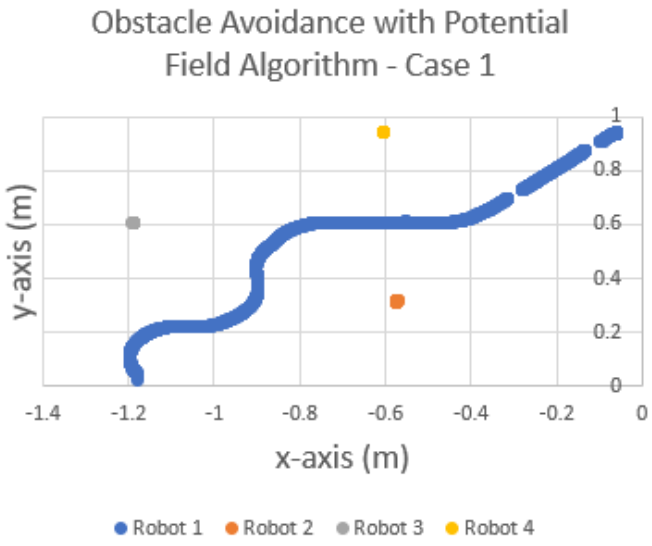

Figure 4.8: Obstacle avoidance result with potential field algorithm - Case one

In Figure 4.9, another navigation result with obstacle avoidance is shown. The starting position of robot 1 was the same with case 1, and the position of robot 4 was slightly moved. The potential field algorithm was able to avoid the obstacles in a dynamic situation, the path generated was different and avoided the obstacles automatically.

The video of the robot motion can be visited through the link: https://youtu.be/DXzEfVXKqEk.

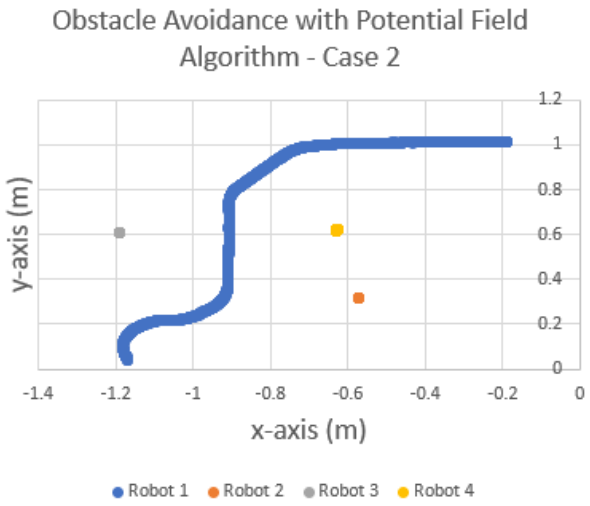

Figure 4.9: Obstacle avoidance result with potential field algorithm - Case two

By comparing the studied algorithms, quintic polynomials algorithm was chosen and implemented to formation control. The main reason was the turning angle calculated by quintic polynomials algorithm was smoother compared to A* algorithm and potential field algorithm. A* algorithm and potential field algorithm would be further studied for real-time obstacles avoidance as the calculations were less complex compared to quintic polynomials equation.

4.3 Distributed Formation Control Results

With the integration of communication, robot navigation and the consensus law (3.20), (3.21), seven formations have been generated to evaluate the system. In these formations, only four robots were used to avoid unnecessary collision. To highlight the flexibility of the formation network, a random graph for four robots and one virtual leader were assigned to each formation as their information topology by using Erdõs-Rényi algorithm, with the probability of 0.6 for each possible link in the graph. In each formation, the graph of the communication network was

first presented, which was then used to calculate the adjacency matrix for the consensus law. Then the expected coordinates or path and the actual path of each formation were presented. Finally, the formation was evaluated using the magnitude of the differences between the actual positions and the desired positions of every agent i:

$$J_i(t) = \left\| \vec{r}_i^{\,d}(t) - \vec{r}_i(t) \right\|, \quad i = \{1, \dots, n\} \tag{4.1}$$

where $r = \begin{bmatrix} x \\ y \end{bmatrix}$ here.

4.3.1 Formation One: Circle Formation

Communication network:

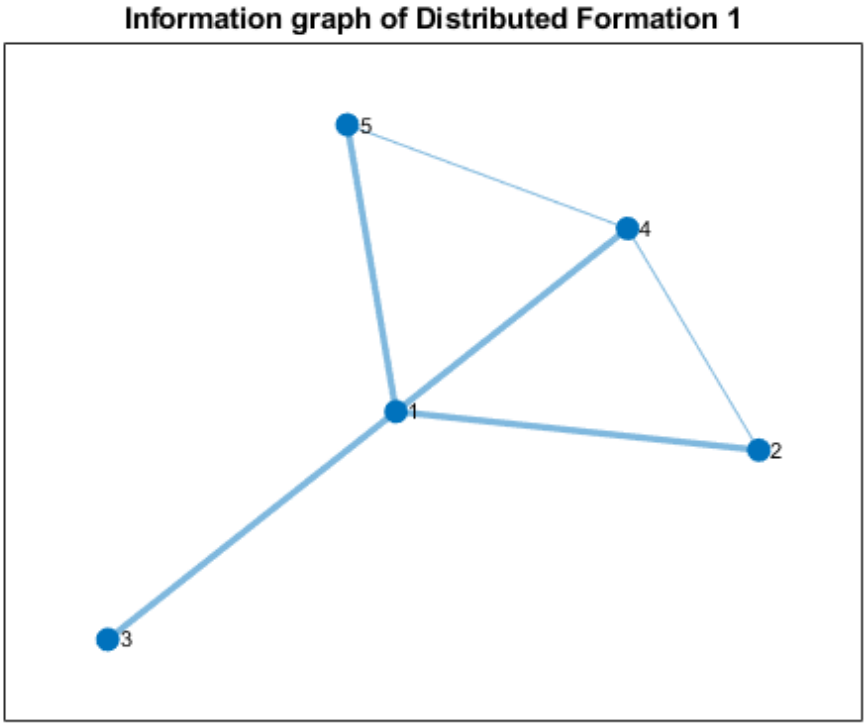

Figure 4.10: Communication network of Formation one

From Figure 4.10, It can be seen that in this information topology, agents 1 and 4 could access the desired formation state $\vec{\xi}_d$.

Expected goal:

This formation was to form a circle around the origin (0,0) from random position:

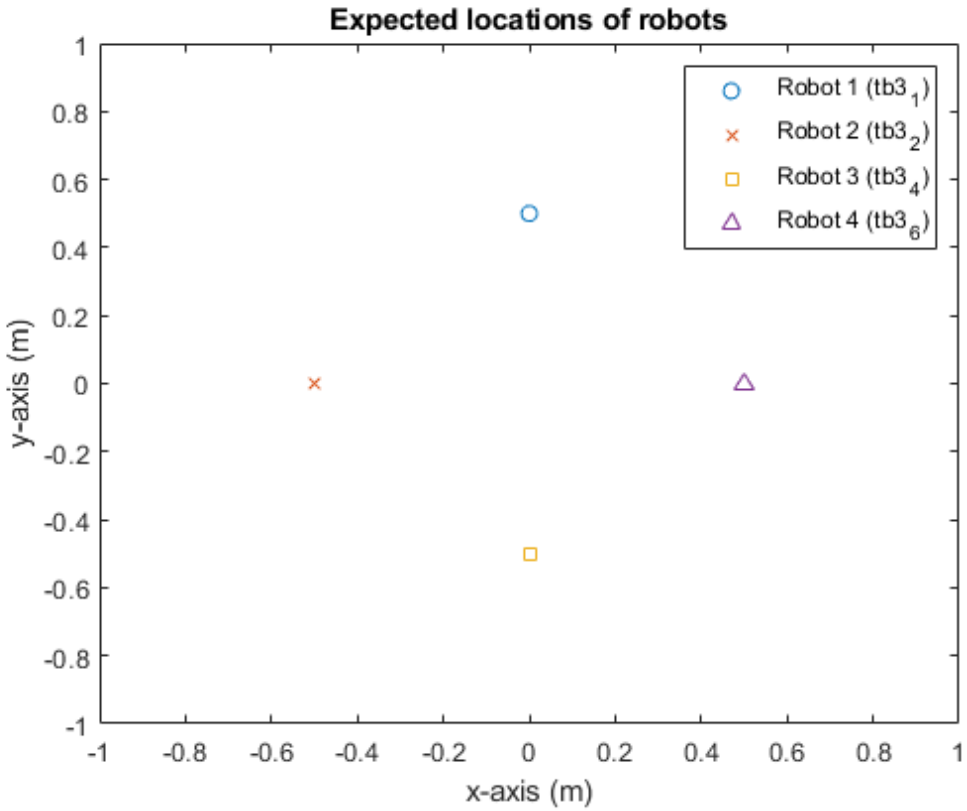

Figure 4.11: Expected robot locations of Formation 1

Actual paths:

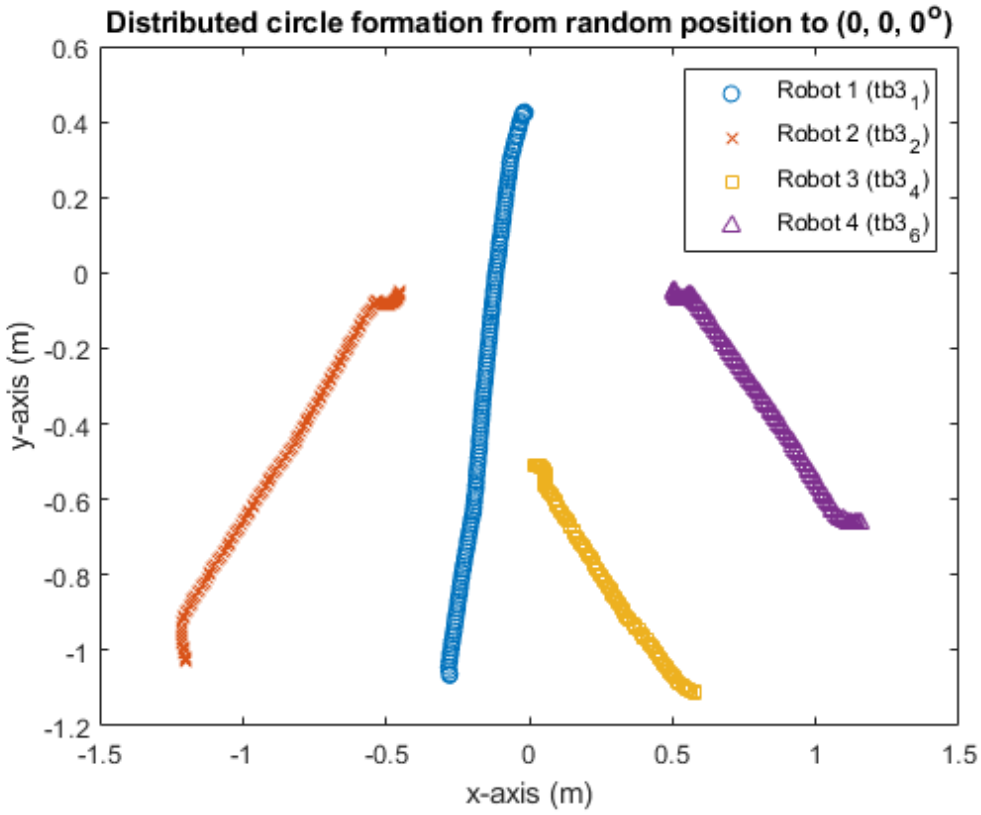

Figure 4.12: Actual Paths of individual robots of Formation one

The video of the robot motion can be visited through the link: https://youtu.be/BdMf-tZKx-U.

Positional error:

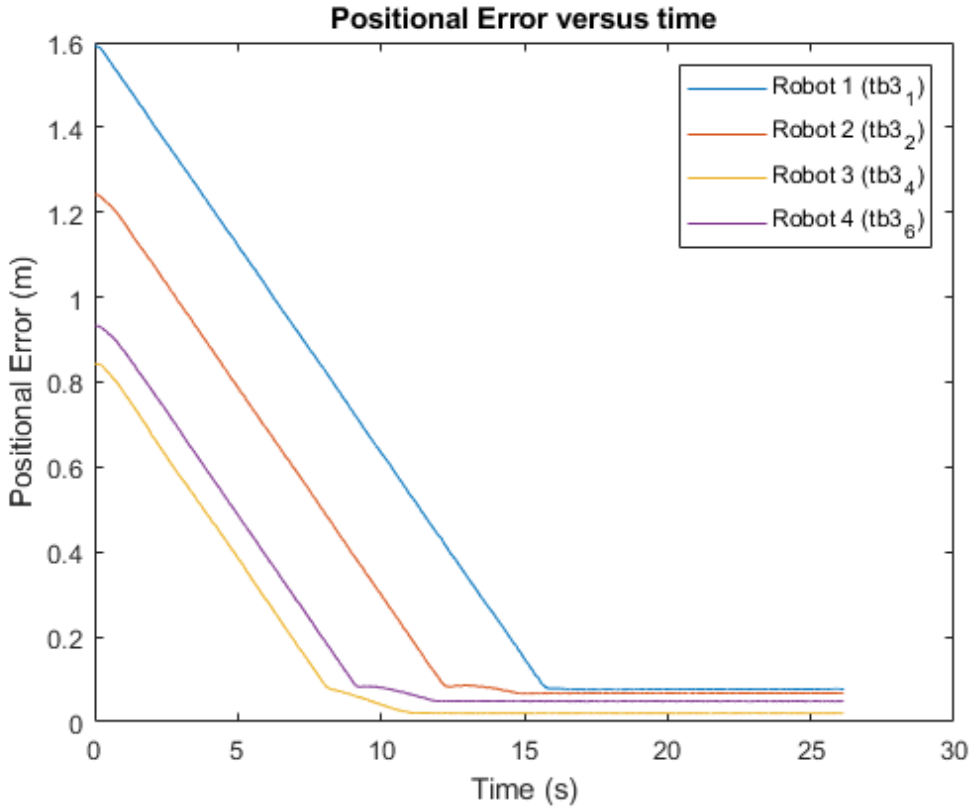

Figure 4.13: Positional Error of Individual robot in Formation one

From Figure 4.13, it is shown that this formation, all the robots reached their goals within the positional tolerance in 16 seconds. Together with Figure 4.12, it is noted that the paths that the robots took to reach their goal position were straight lines, this suggests a good performance in path tracking.

4.3.2 Formation Two: Line Formation One

Communication network:

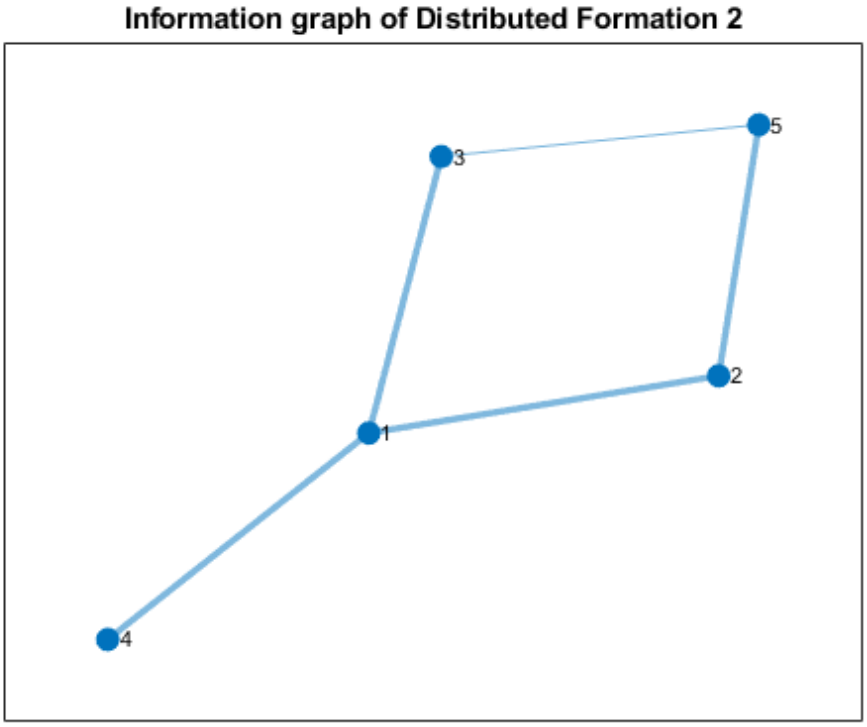

Figure 4.14: Communication network of Formation two

From Figure 4.14, it is shown that in this occurrence, the desired formation state $\vec{\xi}_a$ could also be accessed by two agents which were agent 2 and 3 here.

Expected goal:

In this formation, the robots were expected to form a line along the 30° direction of the point $(0, -0.5)$:

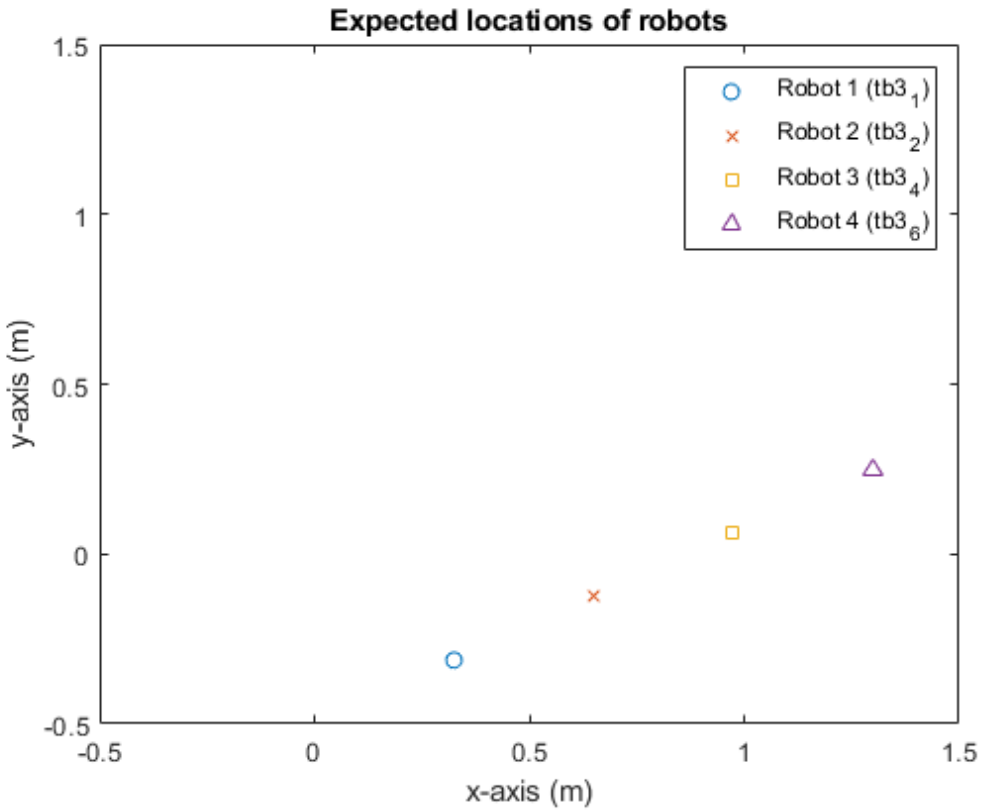

Figure 4.15: Expected robot locations in Formation two

Actual paths:

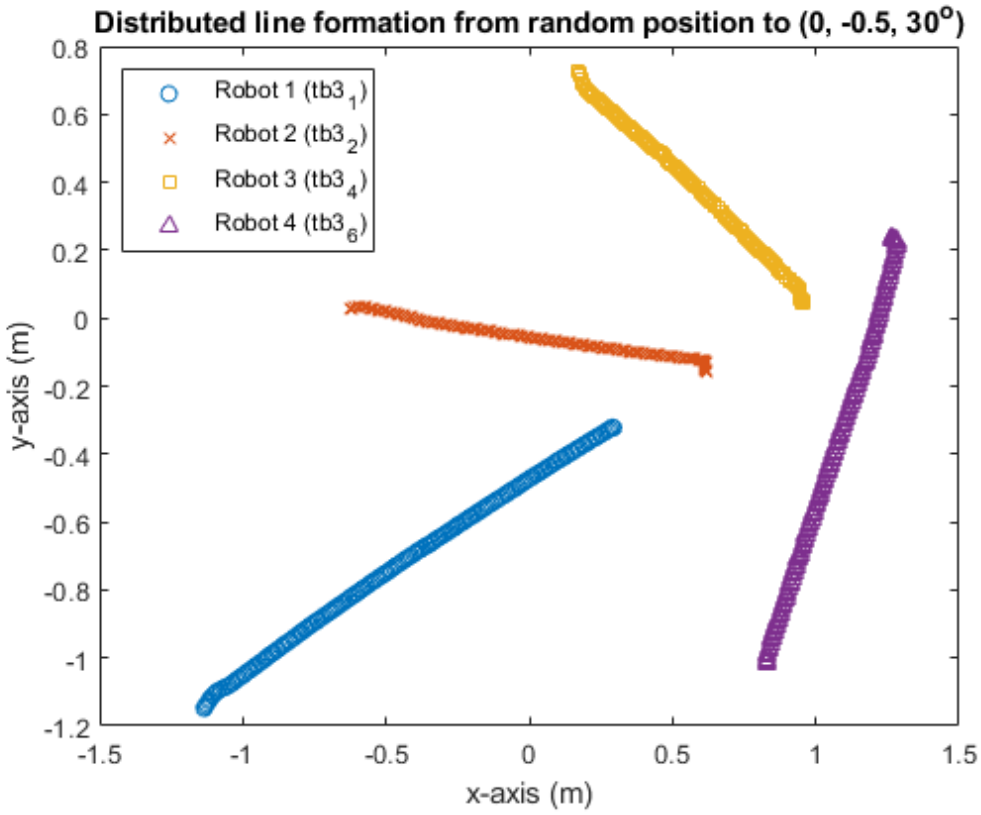

Figure 4.16: Actual Paths of individual robots of Formation two

The video of the robot motion can be visited through the link: https://youtu.be/RkE-h590ZpQ.

Positional error:

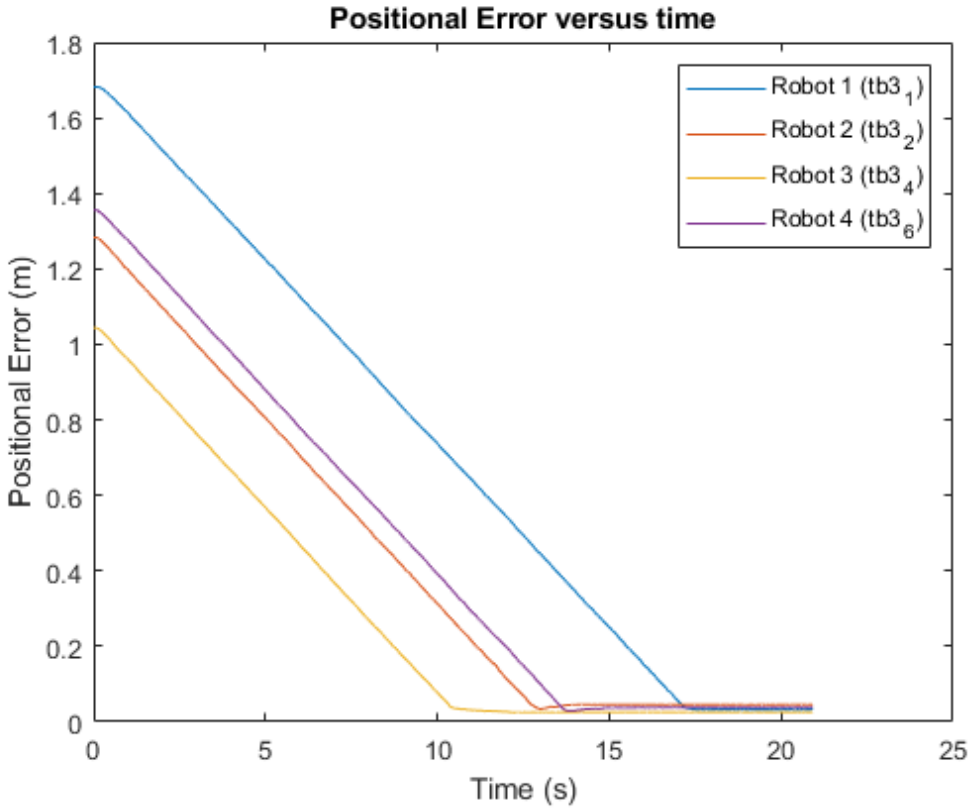

Figure 4.17: Positional Error of Individual robot in Formation two

Figure 4.16 and 4.17 show a similar result to that in Formation one, the robots in this formation also found their way to their respective goal pose directly, without any unnecessary curvature.

4.3.3 Formation Three: Line Formation Two

Communication network:

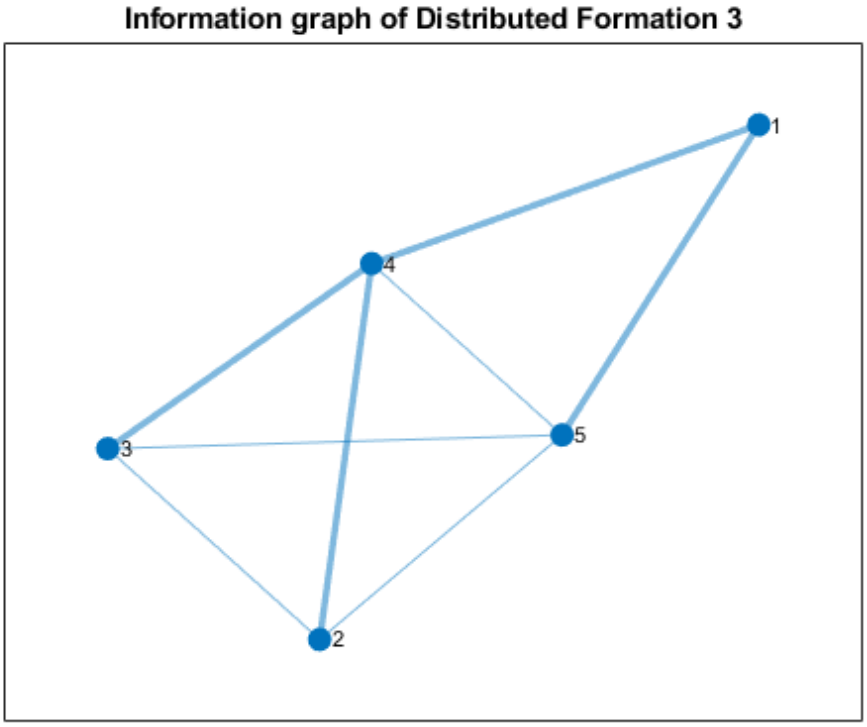

Figure 4.18: Communication network of Formation three

Figure 4.18 shows that in this occurrence, the desired formation state $\vec{\xi}_d$ was known by all four robots.

Expected goal:

In this formation, the line formation in Formation 2 was translated to another line formation at the 0^o direction of the point $(0, -1)$.

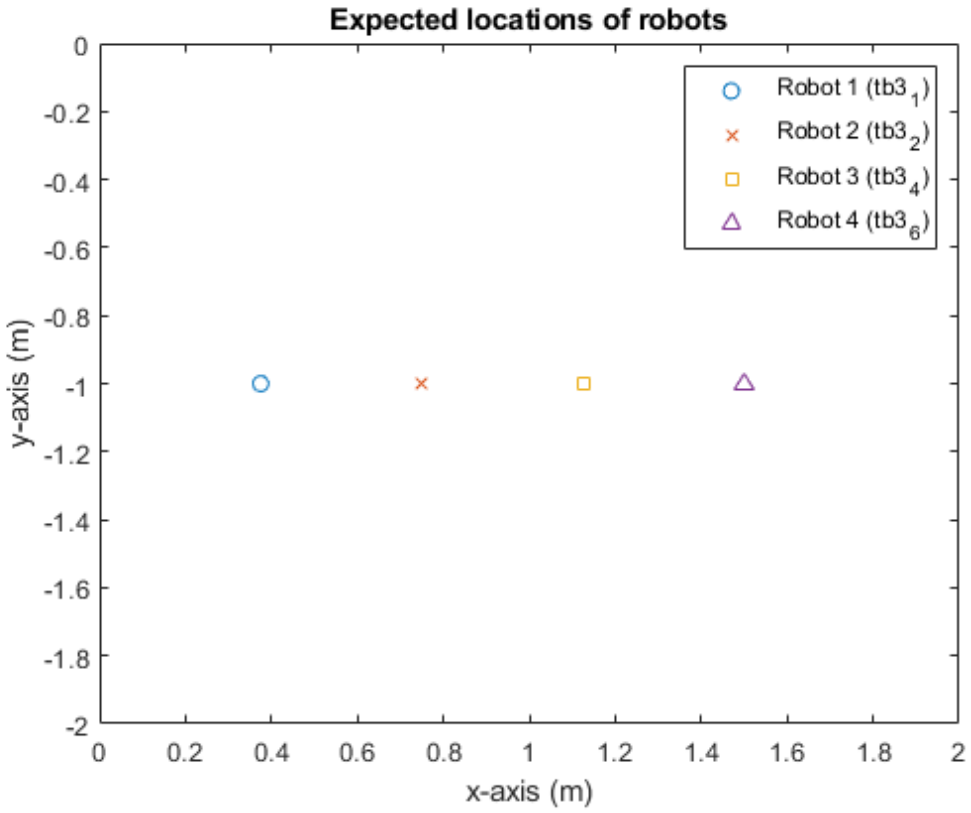

Figure 4.19: Expected robot locations in Formation three

Actual paths:

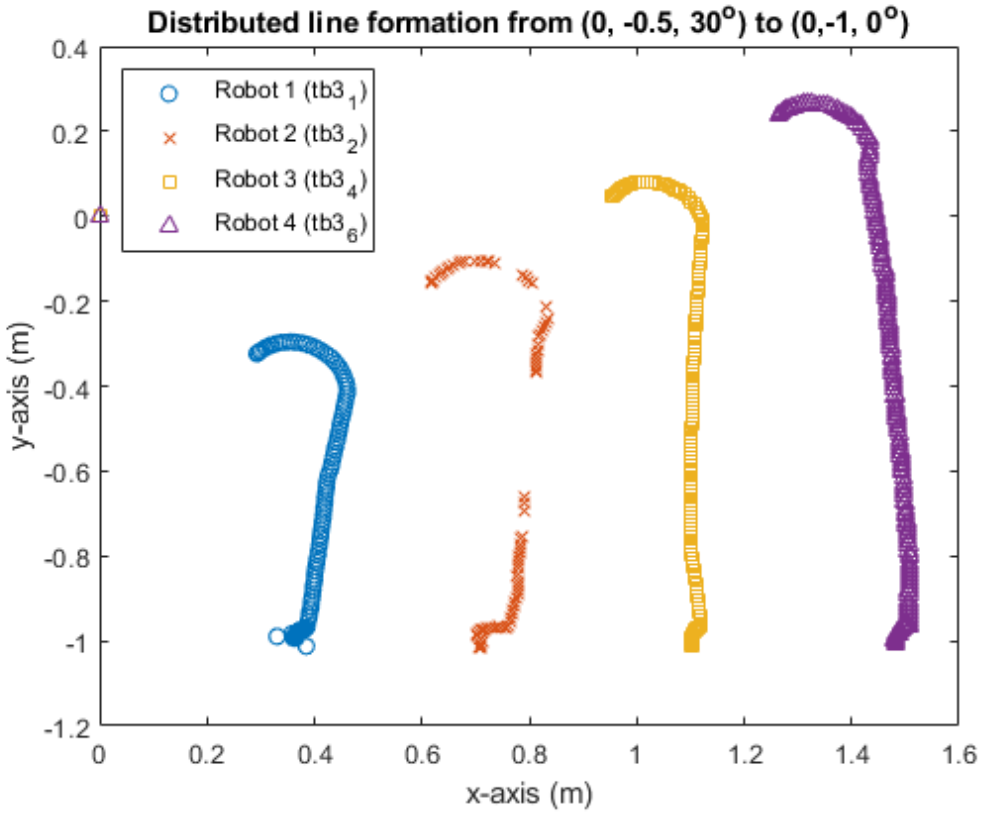

Figure 4.20: Actual Paths of individual robots of Formation three

The video of the robot motion can be visited through the link: https://youtu.be/qRICIuNEZD4.

Positional error:

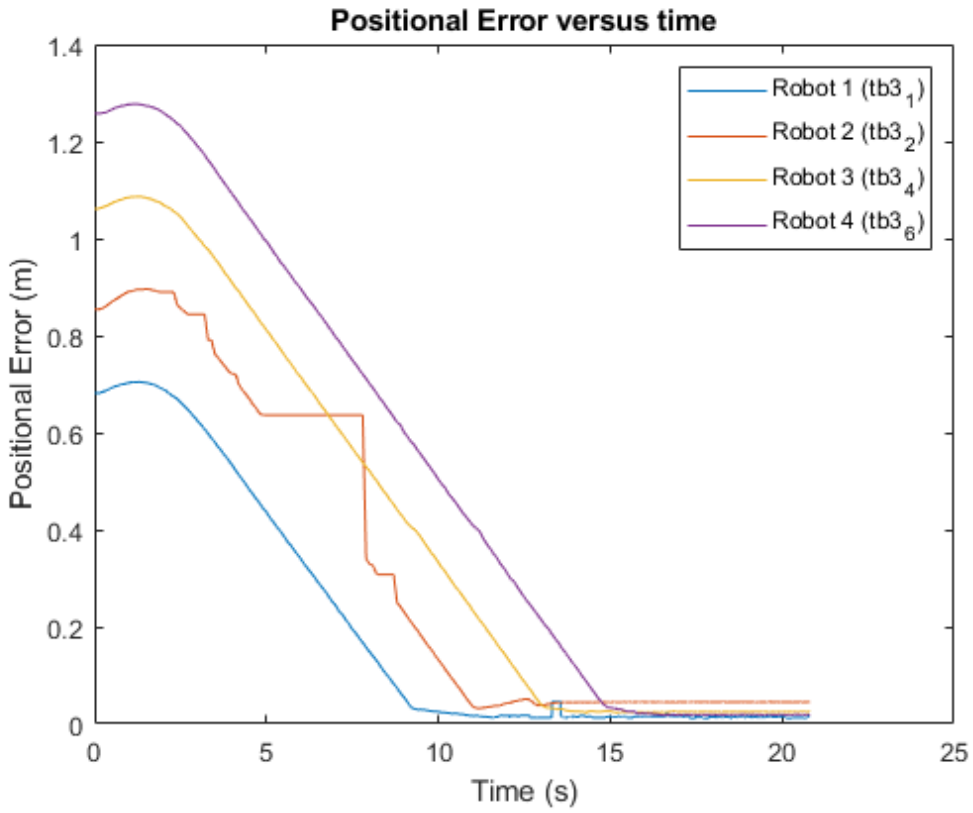

Figure 4.21: Positional Error of Individual robot in Formation three

Figure 4.20 and Figure 4.21 also show a similar result to that of Formation 1. The curved path that every robot turned in the beginning of the formation was as a result of the initial orientation of the robots pointing at 30° with respect to the coordinate $(0, -0.5)$. This suggests an improvement in the path planning section. Since the robots could rotate their wheels separately, the robots could first self-rotate to their goal position before moving to there, but this may only be limited to some applications. For example, this could not be done if the system was applied to a swarm of airplanes. On the other hand, in this formation, notable disconnection of the path of robot 2 was observed, which was also revealed in Figure 4.21 as an abrupt change between 5 to 8 secs for agent 2. This was due to package loss in the OptiTrack motion tracking system, which might ultimately be due to the size limitation of the effective motion capturing region and the wireless communication system. In fact, this happened throughout experiments and this issue will be further discussed in section 4.5.

4.3.4 Formation Four: Rotating Circle Formation

Communication network:

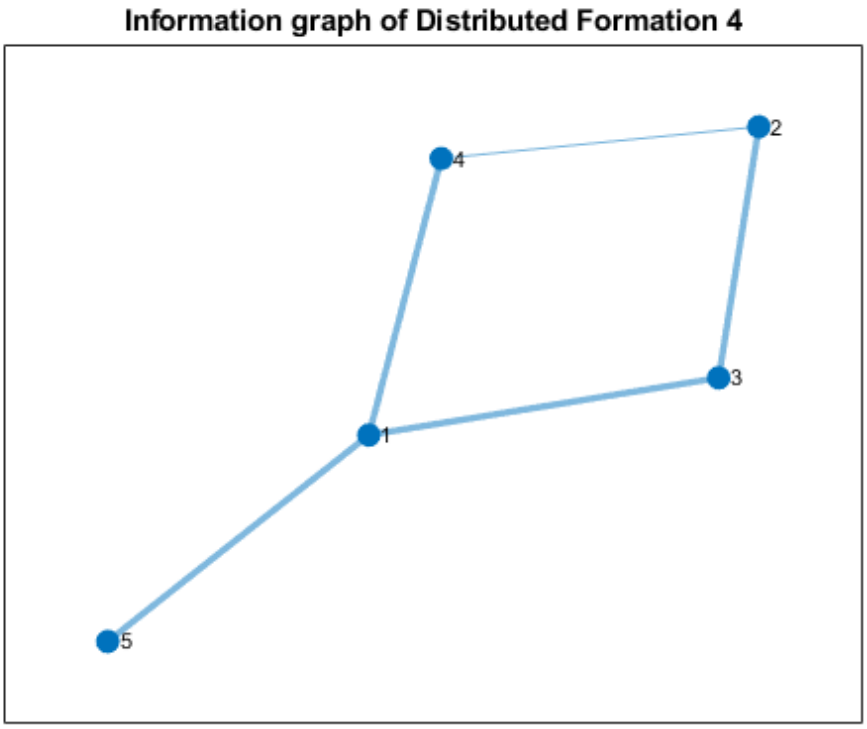

Figure 4.22: Communication network of Formation four

From Figure 4.22, in this occurrence, the generated communication network only allowed agent 1 to know the desired formation state $\vec{\xi}_d$.

Expected goal:

In this formation, time varying formation was conducted instead of time invariant formations like Formation one to three, since the effective space was not large enough, the time varying formations chosen would be centered around a fix point. In this experiment, the robots were expected to follow a circular path around the point $(0, -0.5)$ from random positions.

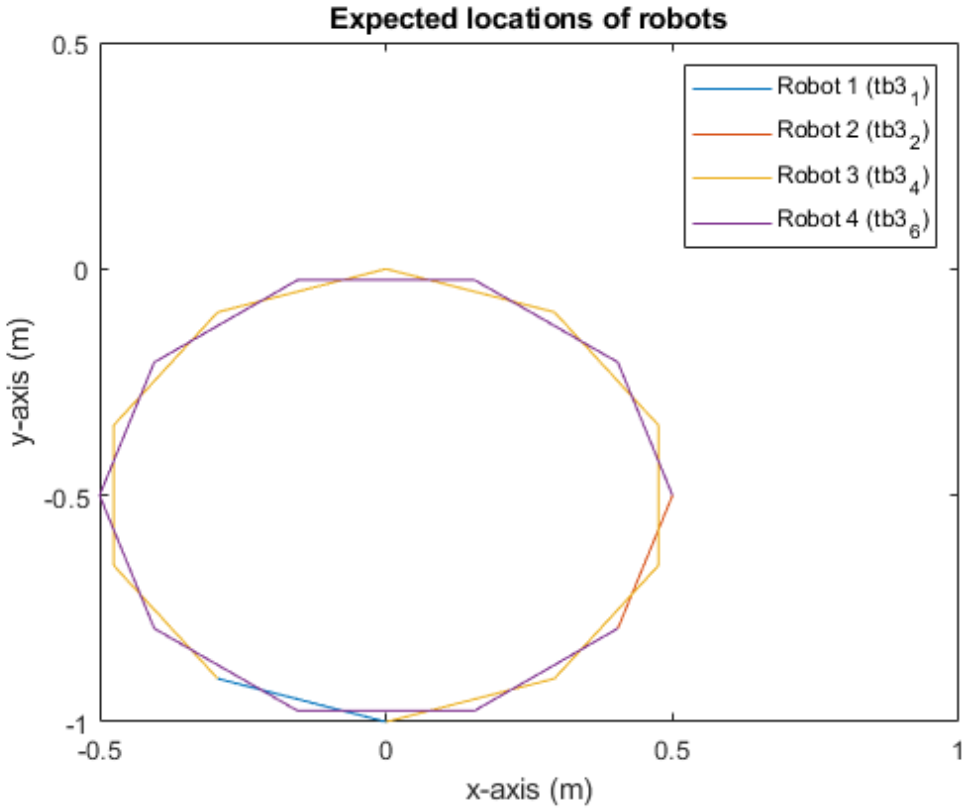

Figure 4.23: Expected robot locations in Formation four

This formation was done by rotating the formation orientation by an angle of $\frac{\pi}{5} rad$, which was, the virtual leader orientation, and successively updated this information at a time interval. In this formation, a 3-second time interval was chosen.

Actual paths:

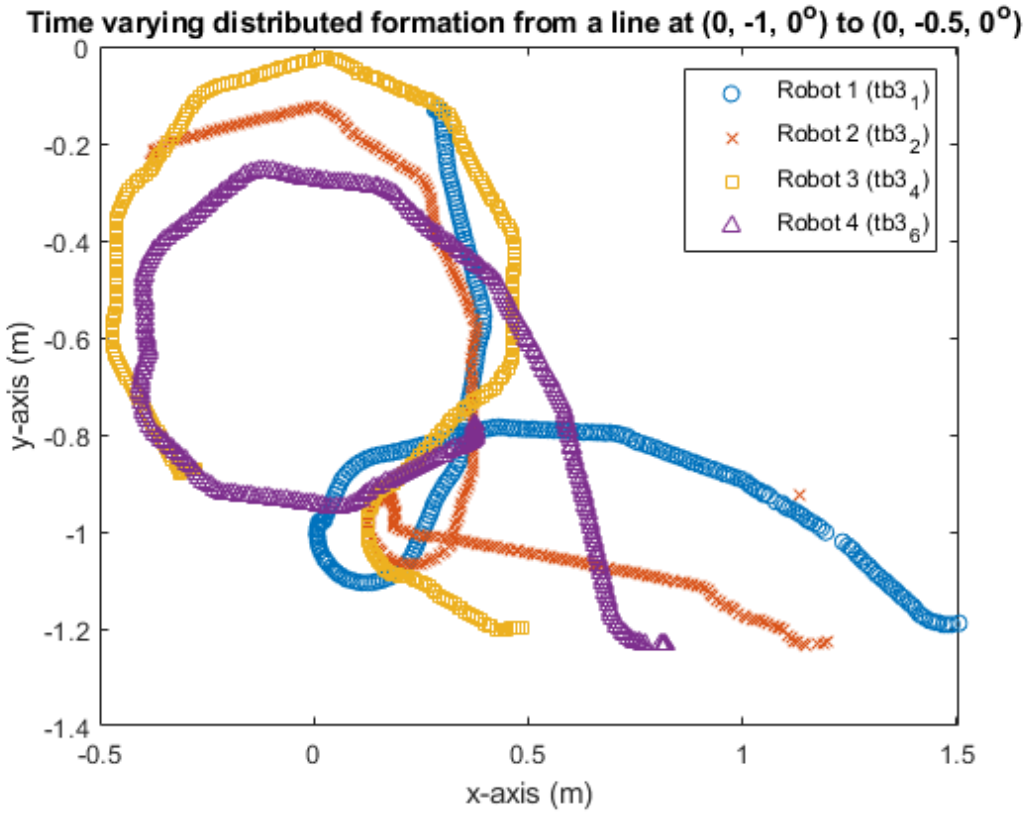

Figure 4.24: Actual Paths of individual robots of Formation four

The video of the robot motion can be visited through the link: https://youtu.be/h3bHgZ5aNi0.

Positional error:

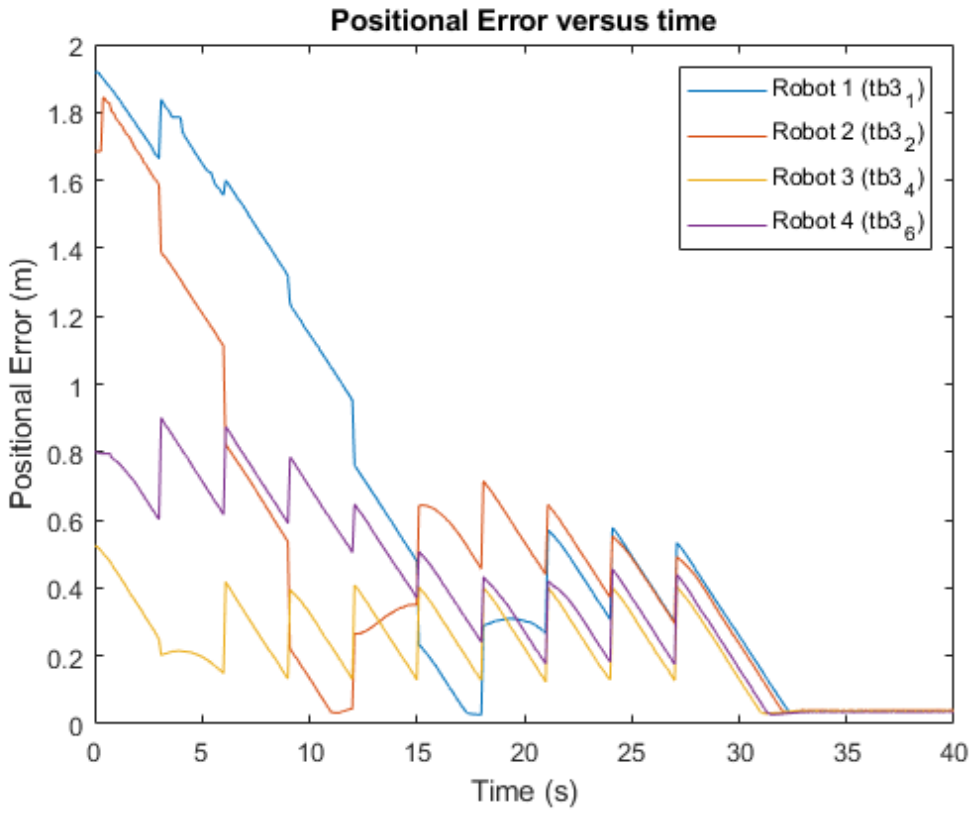

Figure 4.25: Positional Error of Individual robot in Formation four

Figure 4.24 and Figure 4.25 show that all the robots had their positional errors converging to around 0.4m at 15 second until the formation ends. This suggests that the update rate of the goal position is too fast. The spikes in each curve in Figure 4.25 were as a result of the formation pose update. Since the pose updated had a distance from the robots, the positional errors jumped to higher values every 3 seconds.

4.3.5 Formation Five: Rotating Line Formation

Communication network:

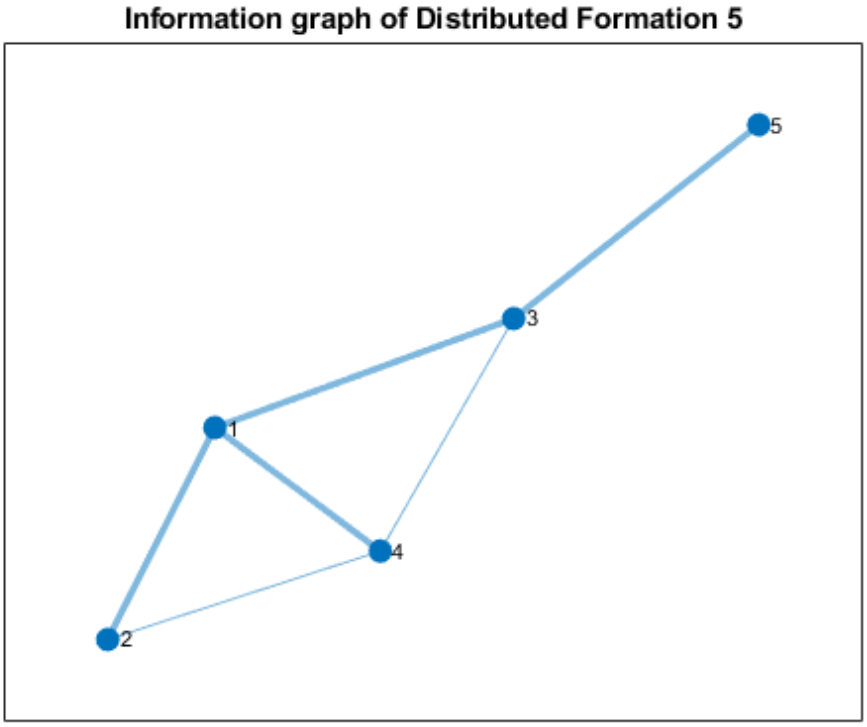

Figure 4.26: Communication network of Formation five

This occurrence of graph generation gives a communication network which only allowed agent 3 to know the desired formation state $\vec{\xi}_d$.

Expected goal:

In this formation, the robots were expected to follow a circular path around $(0,0)$ from $0 \rightarrow \frac{9}{5}\pi\ rad$ as shown.

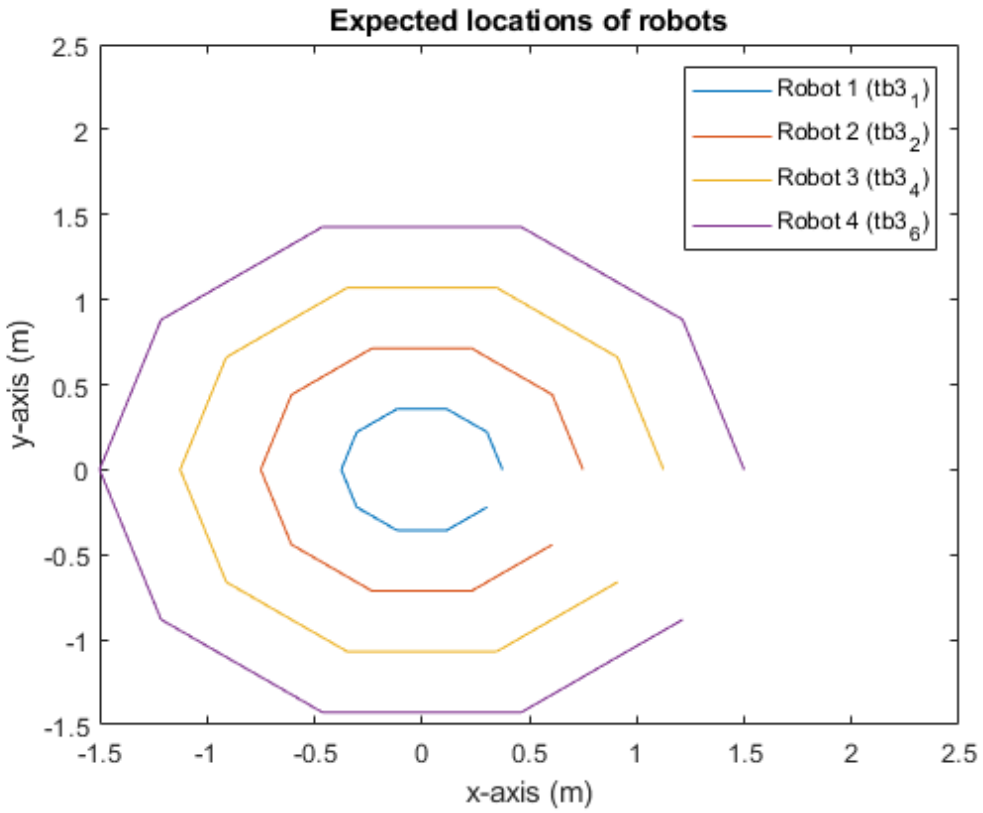

Figure 4.27: Expected robot locations in Formation five

The updating rate was maintained at $\frac{\pi}{5}\ rad$ every 3 seconds in this case. The robots were placed at the end points located along the -30° direction of $(0,0)$.

This time, instead of circulating around the same circle, the robots had distinct circular path around the same coordinate with different radius. This formation could be viewed as a combination of Formation 2 and 4.

Actual paths:

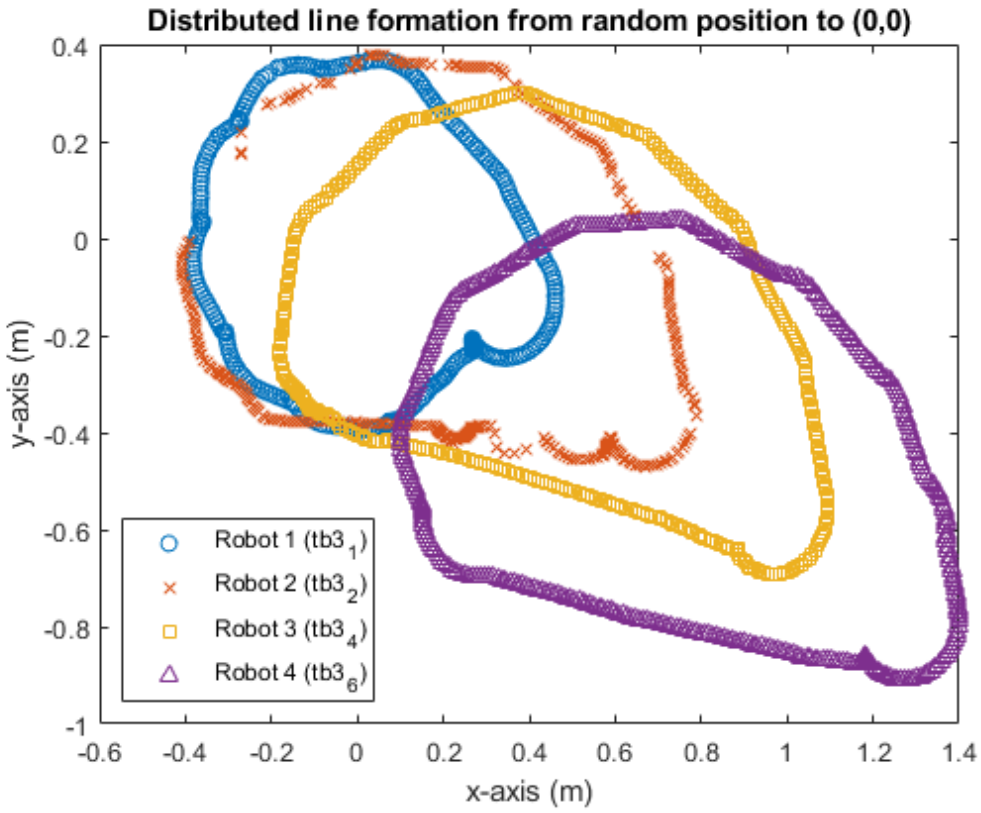

Figure 4.28: Actual Paths of individual robots of Formation five

The video of the robot motion can be visited through the link: https://youtu.be/VMWskVCYGFE.

Positional error:

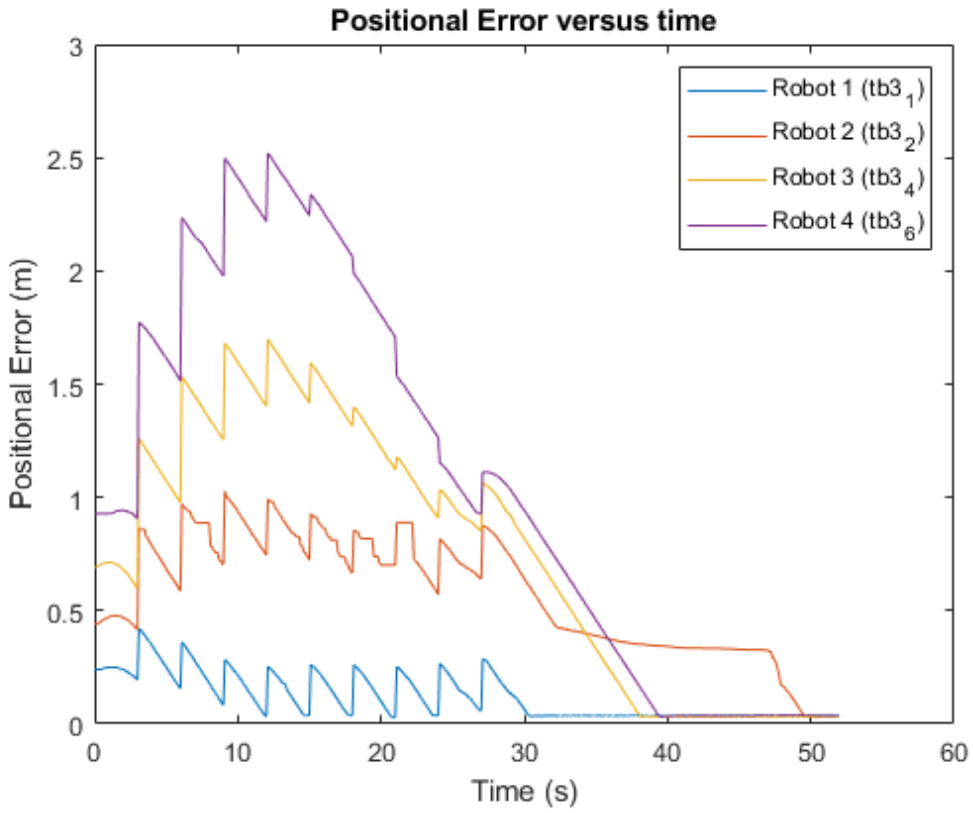

Figure 4.29: Positional Error of Individual robot in Formation five

From Figure 4.28, it can be seen that there were 4 distinct closed paths. They were closed because the robot started at the final goal positions of the formation. The figure also reveals that the robots tried to follow the prescribed paths but failed to do so. This is a result of constant velocities. Since the velocities of all the four robots were constant, they could not catch up with the other robots to form radical lines at the checkpoints. This phenomenon worsened for agents at the outer rim as shown in Figure 4.29, agent 4 had an error of 2.5m, the highest positional error in the middle of the formation, and agent 1 had the smallest positional error at all times. This suggests that a future development is necessary to eliminate this high positional error by controlling the acceleration of each agent, the agents at the outer circle should have a higher acceleration. This will help to maintain the integrity formation.

4.3.6 Formation Six: Spiral Formation

Communication network:

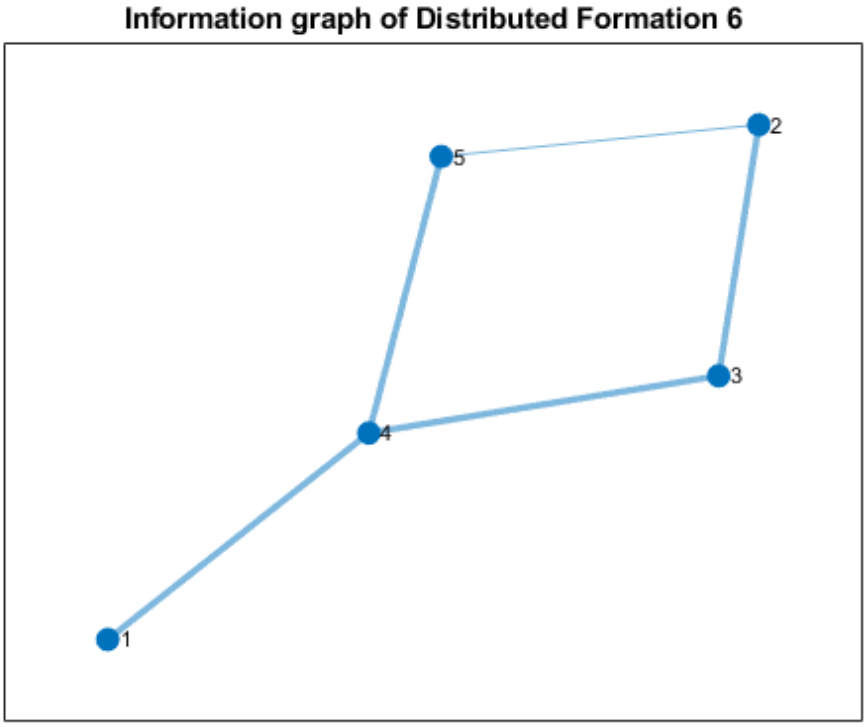

Figure 4.30: Communication network of Formation six

In this formation, the desired formation state $\vec{\xi}_d$ could be accessed by agent 1 and 2 only.

Expected goal:

In this formation, time varying formation with spiral pattern was explored. Starting from this formation, a varying radius into the formation was introduced. In this experiment, the four robots were first located at $(0,0.5)$, $(-0.5,0)$, $(0,-0.5)$, $(0.5,0)$, then spiraled out by adding an increasing radius of $0.1m$ every 3 seconds.

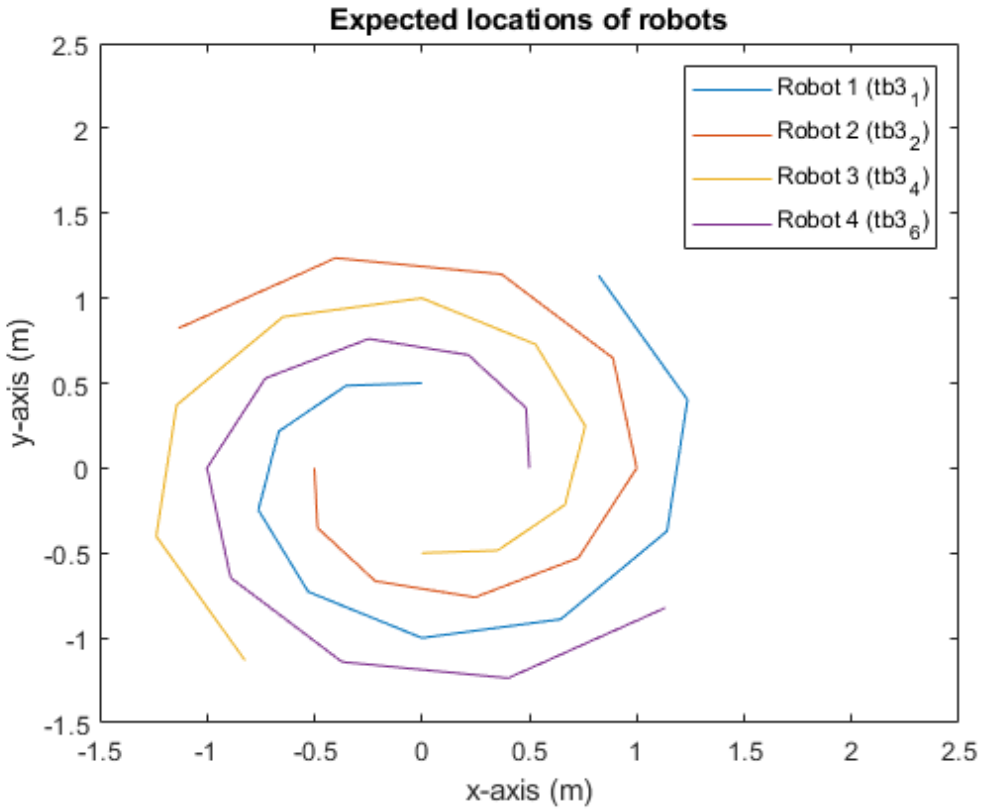

Figure 4.31: Expected robot locations in Formation six

Actual paths:

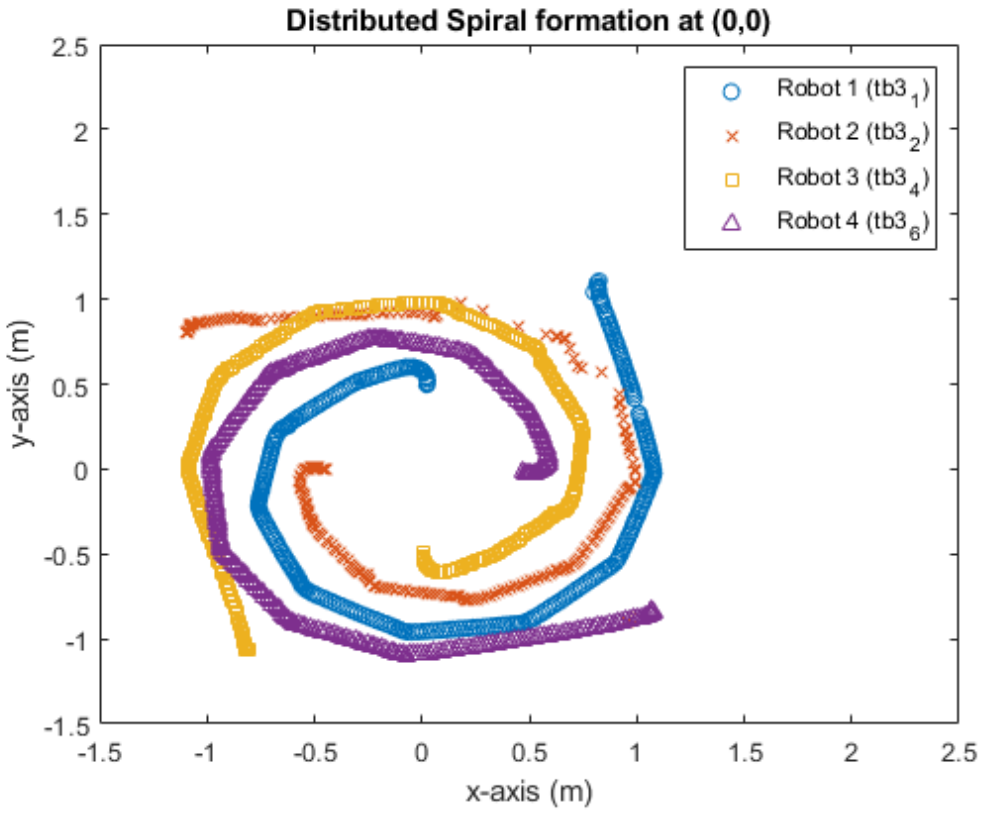

Figure 4.32: Actual Paths of individual robots of Formation six

The video of the robot motion can be visited through the link: https://youtu.be/xbunvrnfAhI.

Positional error:

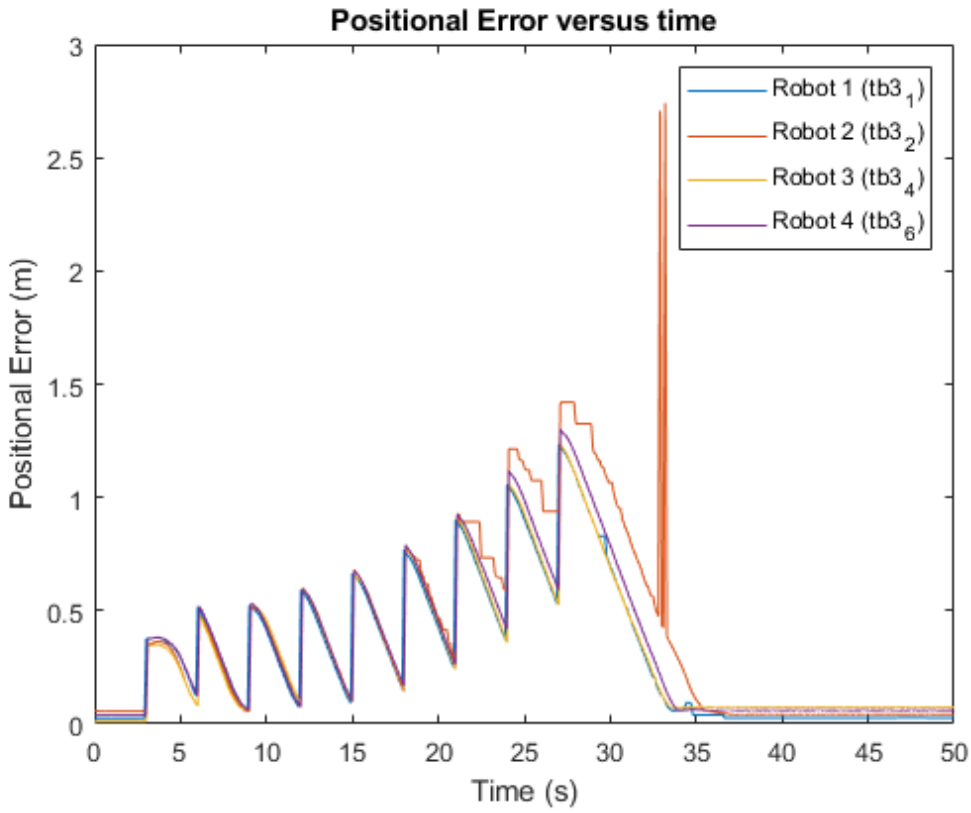

Figure 4.33: Positional Error of Individual robot in Formation six

Figure 4.32 shows that the robots kept tracking the paths with some deviation. As shown in Figure 4.33, the position error kept increasing until the second last checkpoint of each agent. This is a result of the increasing length of the path between each checkpoint and the constant update rate. At longer paths, when the formation was forced to update, the robots might still be distant from the checkpoints they were heading to. The robots then subsequently changed their goal positions to the new ones, resulted in an increasing trend of positional error. This suggests another future development may be necessary for a time varying formation update rate for formations with varying radius and orientation around a fixed point to maintain the formation shape.

Moreover, same as the last formation, the spikes in Figure 4.33 represent formation pose updates except the one at 34 seconds, which was cause by the package loss issue raised in Formation 3.

4.3.7 Formation Seven: Flower Formation

Communication network:

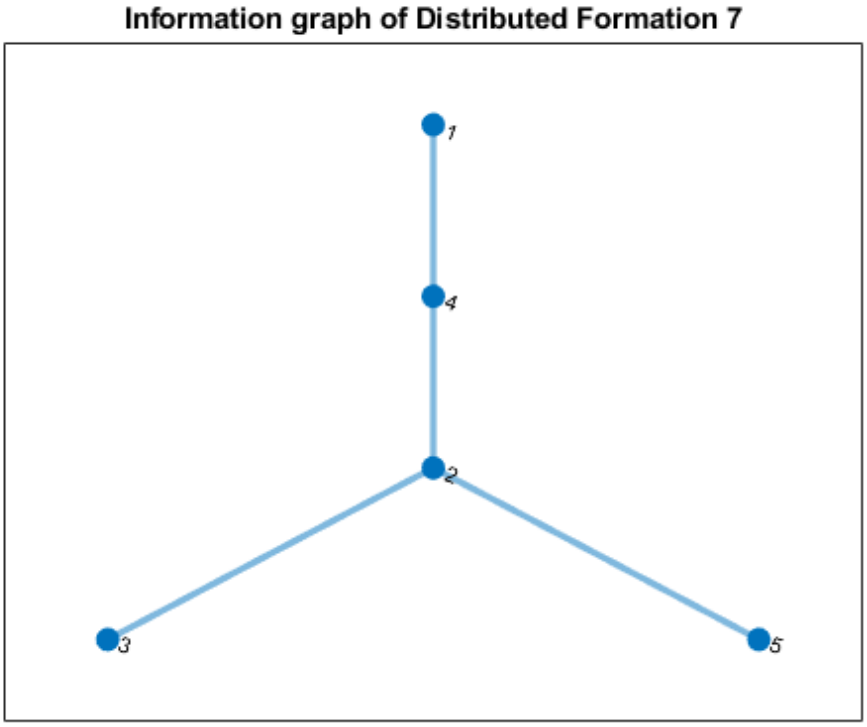

Figure 4.34: Communication network of Formation seven

In this occurrence, only agent 2 was allowed to access the desired formation state $\vec{\xi}_d$.

Expected goal:

The last formation tested the formation system with an increasing radius and raised the concern of time varying formation update rate. In this experiment, the importance of the time varying formation update rate was highlighted in varying radius and orientation formations around a fixed point by introducing a flower shape formation with radius r as a function of the formation orientation θ in the form of:

$$r = |\sin(n\theta)| + \frac{1}{2} \tag{4.2}$$

where n is the number of robots. In this case, there were 4 robots, so $n = 4$.

This means the radius of the formation would be at least $0.5m$, and from that on a varying additional radius $0 \leq |\sin(4\theta)| \leq 1$ would be imposed on the formation, which depended on the formation orientation θ.

This formation had some sharp changes in direction for every 45°. With the idea of eliminating idle time between short paths, this formation was updated at a rate of $\frac{\pi}{10}\ rad$ per 2 seconds, which resulted in the formation path below:

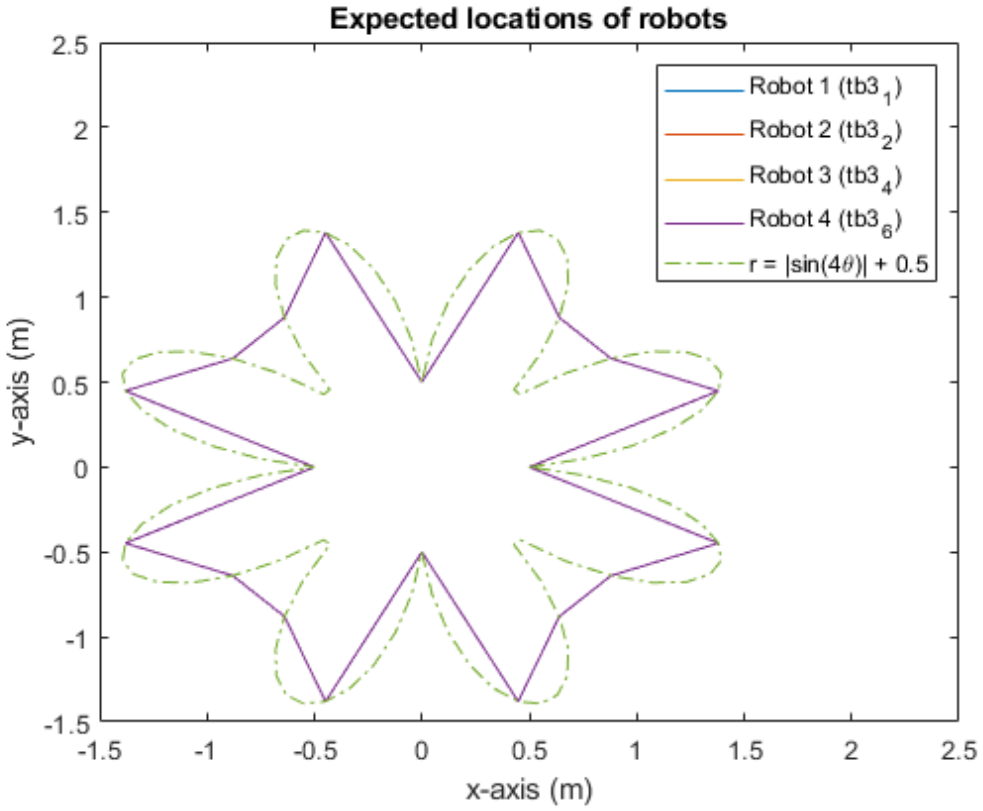

Figure 4.35: Expected robot locations in Formation seven

From Figure 4.35, it can be seen that there are sharp changes in direction at 0°, 90°, 180°, 270° with respect to the coordinate (0,0) in the formation path, and the paths connecting these four points were relatively longer than other paths.

Actual paths:

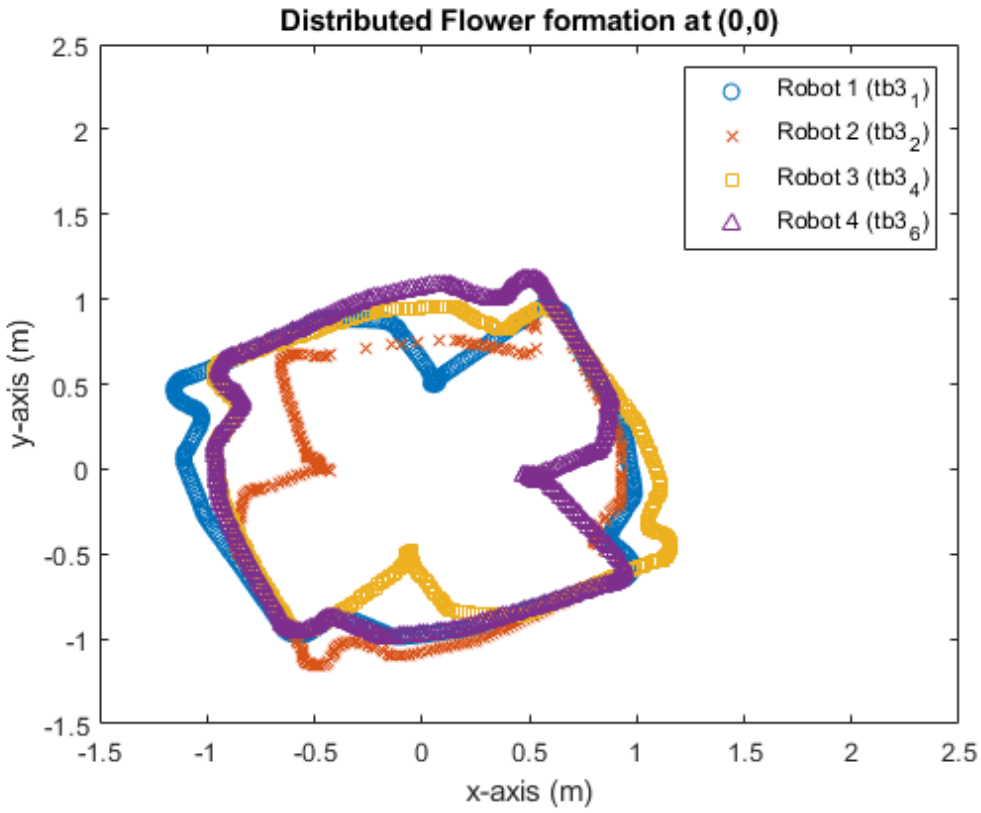

Figure 4.36: Actual Paths of individual robots of Formation seven

The video of the robot motion can be visited through the link: https://youtu.be/Gru88MXBDaI.

Positional error:

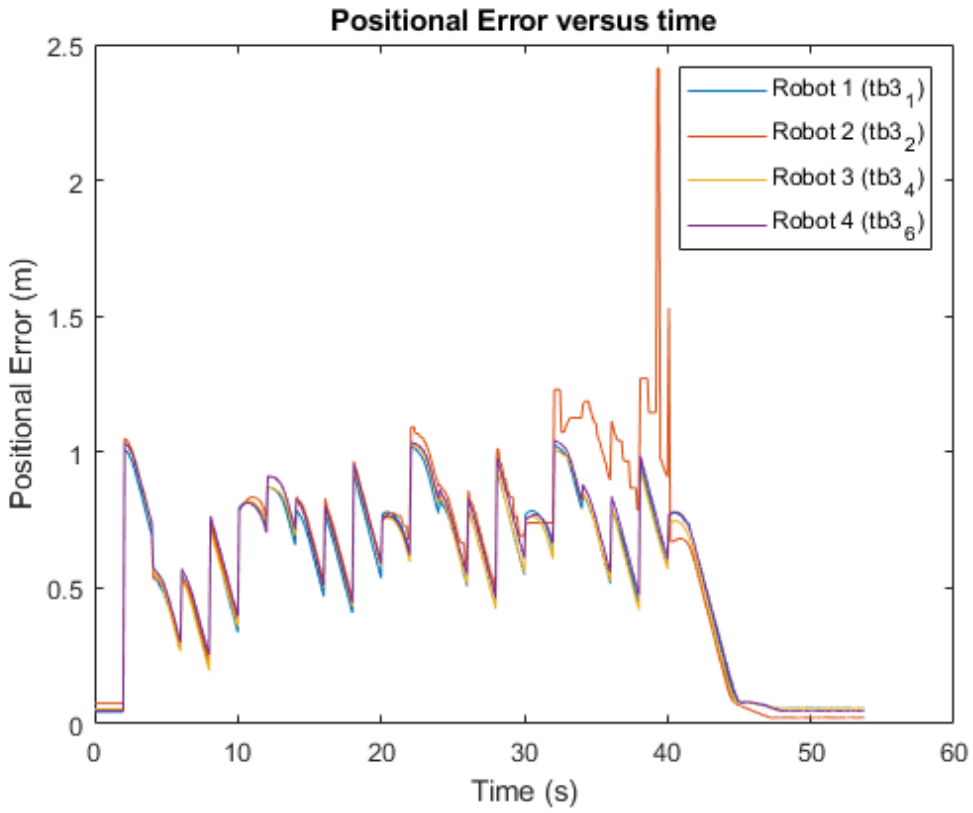

Figure 4.37: Positional Error of Individual robot in Formation seven

Figure 4.36 shows the actual paths of each robot in this formation, it can be seen that every robot was found to have difficulties in turning into the sharp angles at 0°, 90°, 180°, 270° of the formation except the last checkpoint. This again suggests the development in time varying update rate for varying radius and orientation formation around a fix point. Again, the spike at around 40 second was due to the package loss issue.

4.4 Limitations and Difficulties Encountered

4.4.1 OptiTrack: Missing Packages

During the development of experiments of the study and implementation of the formation scheme, the following difficulties had been encountered.

Recall Figure 4.4, a portion of path near (-0.1, 0.6) is missing dots which represent the loss of feedback and communication information packages from OptiTrack. At the first application, this missing feedback caused the program to remain at the feedback stage of the program and the robot would move in an uncontrolled manner. The ability of path tracking of the robot was lost. To deal with issues engendered by the missing packages, adjustments were made to the program structure. These adjustments helped resolve the limitations as regards OptiTrack.

However, since feedback is lost when there are missing packages from OptiTrack, feedback control could only be conducted by adopting feedback estimations. The effects of missing of packages could then be diminished.

4.4.2 Varying Control Frequency in ROS

During the experiments of the path planning, path tracking and distributed formation control, the control frequency was set as 10 Hz. However, the actual frequencies varied from 8 Hz to 10 Hz corresponding to the computation time in each sample. When the computational load increased at the instants due to heavy calculations like obstacle avoidance, the time for calculations increased and the control frequency dropped. This phenomenon is related to the architecture of ROS which could not be solved in the program used in this study. The inconsistent control frequency could lead to unstable motion performances of the robots and increased the difficulties in predicting the future performances of the robots.

4.4.3 Difficulties in real-time Obstacle avoidance

The information received by each robot is restricted. A robot can only receive its own position feedback and the neighbor's position feedback. In an identical robot's path planning, the current method to update the obstacle map is to use the neighbor's position from OptiTrack motion capture system, and the actual size of robot to set the corresponding coordinates as obstacle. As a result, both A* path planning algorithm and potential field algorithm can treat the neighbor robots as obstacles and the calculated path will avoid collisions between the robots and its neighbors.

Therefore, when real-time updating the obstacle graph in each robot, the calculated optimal path may intersect with other robot which is not the neighbor. As a result, the path planning algorithm currently can only ensure not to collide with the robots' neighbor, unless additional information is received by the robot.

4.5 Summary

In this chapter, results of path tracking and path planning were first introduced. To link the path with time, combination of path planning and path tracking was adopted. The results of distributed formation control are also presented with seven formation cases. Limitation of losing packages of OptiTrack and difficulties in real-time Obstacle avoidance are also discussed.

CHAPTER 5

CONCLUSION

5.1 Objectives Recap

There are four main objectives in this study. The first objective is to investigate and perform navigation of single UGV with path planning, path tracking and obstacle prevention. The second objective is to investigate different formation and consensus algorithms commonly adopted in researches. The third objective is to perform distributed formation control of multiple UGVs with time-varying formation patterns. The final objective is to evaluate the feasibility and performance of a virtual leader approach of formation control scheme using a low cost distributed computational system.

5.2 Major Findings

The distributed formation scheme was validated by the results of seven formation experiments where the robots were able to form target patterns with consensus. Although experiment one to three shows satisfactory results, experiment four to seven shows difficulties for the robots to follow the formation paths, which suggests future developments in acceleration limits, time-variant update rate of formation, time-variant communication network of the formation, and formation rearrangement. These will be further discussed in Chapter 6.

Some limitations and difficulties were found which include limitation of lose packages of OptiTrack and difficulties in real-time obstacle avoidance.

5.3 Conclusions

In this study, the literatures of graph theory, consensus and formation control algorithms were reviewed. The optimal formation control algorithm for this study is found to be virtual leader approach due to its easiness of understanding and implementation. The information flowed between neighbors of Multi-agent Systems with the assistances of Graph Theory. Then, the common goals of the agent were aligned by consensus algorithms.

Moreover, the concepts and methodology of robot dynamics were discussed with relevant mathematics equations. The methodology of robot dynamics was then validated by the results of path tracking and path planning experiments of TurtleBot3 Burger. Three types of path planning

were investigated which include Quintic polynomials planning, A* algorithm and potential field algorithm.

In general, the objectives of this study were achieved. Some future works have to be conducted to overcome the limitations stated in this study to polish the distributed formation scheme. This study has proven the possibility of distributed formation scheme on real navigations of UGVs. As the demand of autonomous vehicles is expanding, the distributed formation scheme introduced in this study could be a plausible solution in controlling large scale autonomous vehicles in near future.

CHAPTER 6

FUTURE WORK

6.1 Pose Estimations

From Section 4.4.1, the issue of missing packages of OptiTrack has limited the effectiveness of feedback control in this study. As the decoding package would transfer either a null or previous position message for computation, the kinematics and dynamics for control use are subjected to a drastic change at that instant. This induces an unnatural behaviour to the robots and the rectification is even harder on robots with higher hierarchies in formation, which could possibly affect their child agents.

Therefore, pose estimations need to be adopted in order to maintain feedback control of the robots, thus maintaining the control strategies and stability. Some common approaches to pose estimations include Kalman filter [47] and deep learning. The future poses of the robots can be predicted with the historical data. By adopting pose estimations, the limitation of losing packages of OptiTrack can be minimized.

6.2 Real-time Obstacle Avoidance

From section 4.4.2, real-time obstacle avoidance could not be obtained as the limitation of position information received by robots. Therefore, the additional information different from OptiTrack position feedback is essential for obstacle avoidance with the defined robot neighbor. For instance, the Lidar on turtlebot or even computer vision can be used as addition information to avoid the obstacles in the motion.

To obtain real-time obstacle avoidance in the future, the path planning algorithm's cost function is required to be modified and the study of game theory is essential for the selection of suitable paths. For instance, when two different robots are moving to the same obstacle, they may turn in the same direction and collide with each other before the next calculation is processed.

6.3 Acceleration Limits

The current control model does not consider the acceleration of each robot. Therefore, when the user input formation radius is too large, the velocity of the robots may be too low to reach the target position with the control model currently used. To improve the situation, the control model will need to consider the acceleration of each robot in the future. When the turning angle or distance is too great, the robot is required to accelerate and obtain the formation within a time interval. Also, the update rate should vary with the acceleration profile of robots to obtain a better formation result.

6.4 Time-variant Update Rate of Formation

In the current situation, the time-variant update rate is user-defined in the study. This is due to the maximum velocity of turtlebot being 0.22 m/s and the maximum rotational velocity is 2.84 rad/s, therefore, the current user-defined update rate was found by experiment. In the future, the consensus and formation calculation will need to consider the maximum speed of the robot and calculate the time-variant update rate automatically instead of manual input by the user.

6.5 Motive Updates

While the image processing and position streaming software, Motive 1.8, is native to the camera hardware setup used by the study, the developer of the system has issued an updated software Motive 2.2. The updated version streams the position data using a new NatNet protocol which is not compatible with the decoding ROS package. It is therefore suggested future development should take this into consideration.

Official NatNet SDK does not natively support Unix-based system, the decoding package must be updated or having the master PC running on a ROS framework on a Windows-based environment such as ROS2 beta. Communication problems were not solved for multiple robot streaming in the early stages of the study as the documentation for NatNet protocol was not published by the developer.

6.6 Time-variant Communication Network of Formation

The communication network of the current formation remains the same throughout the formation process. This raised the concern of the situation when the only selected leader disconnects from the network, if this arises, the other agents will not be able to maintain the formation anymore. A solution to this is to impose a time varying communication network to reduce the chance such that the communication network will change from time to time throughout the formation. This would reduce the chance to disconnect a robot which had the only access to the formation state.

6.7 Formation rearrangement

Based on the situation in 6.6, if the formation has a following agent which has been disconnected, the formation will not be completed because there will be a missing agent in the formation. To circumvent this, a camera can be used to highlight a region of interest and consider only the agents inside this region. Then positions of the agents can be rearranged in a way that is suitable for this number of agents.

REFERENCES

[1] A. Hernandez, C. Copot, J. Cerquera, H. Murcia and R. D. Keyser, "Formation Control of UGVs using an UAV asRemote Vision Sensor," *IFAC Proceedings Volumes,* vol. 47, no. 3, pp. 11872-11877, 2014.

[2] L. Yongshen, Y. Xuerong, Y. Yajun and P. Shengdong, "Formation Control of UGVs Based on Artificial Potential Field," in *2018 37th Chinese Control Conference (CCC)*, Wuhan, China, 2018.

[3] Y. Qi, Y. Kang, S. Zhou and S. Yan, "Formation Control for Unmanned Aerial Vehicles with Directed and Switching Topologies," *International Journal of Aerospace Engineering,* vol. 2016, no. 2, pp. 1-8, Oct. 2016.

[4] H. Liu, Y. Tian, F. L. Lewis, Y. Wan and K. P. Valavanis, "Robust Formation Control for a Team of Satellites," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, Dallas, TX, 2018.

[5] T. Ikeda, T. Mita and M. Yamakita, "FORMATION CONTROL OF AUTONOMOUS UNDERWATER VEHICLES," *IFAC Proceedings Volumes,* vol. 38, no. 1, pp. 666-671, 2005.

[6] A. Ajorlou, A. Momeni and A. G. Aghdam, "A Class of Bounded Distributed Control Strategies for Connectivity Preservation in Multi-Agent Systems," *IEEE Transactions on Automatic Control,* vol. 55, no. 12, pp. 2828-2833, 2010.

[7] X. Chen and R. W. Brockett, "Centralized and Decentralized Formation Control With Controllable Interaction Laws," in *Proceedings of the IEEE Conference on Decision and Control 2014*, Los Angeles, California, USA, 2014.

[8] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus and M. H. A. Jr., "Perception, Planning, Control, and Coordination for Autonomous Vehicles," *Machines,* vol. 5, no. 1, 2017.

[9] H. Bai, M. Arcak and J. Wen, Cooperative Control Design: A Systematic, Passivity-Based Approach, Springer Science & Business Media, 2011.

[10] X. Dong, Formation and Containment Control for High-order Linear Swarm Systems, Springer, 2017.

[11] H. Ichihara, S. Kajihara, Y. Ebihara and D. Peaucelle, "Formation Control of Mobile Robots Based on Interconnected Positive Systems," *IFAC-PapersOnLine,* vol. 50, no. 1\, pp. 8441 - 8446, 2017.

[12] H. Du, W. Zhu, G. Wen, Z. Duan and J. Lü, "Distributed Formation Control of Multiple Quadrotor Aircraft Based on Nonsmooth Consensus Algorithms," *IEEE Transactions on Cybernetics,* vol. 49, no. 1, pp. 342 - 353, 2019.

[13] G. L. Qiuling Jia, "Formation Control and Obstacle Avoidance Algorithm of Multiple Autonomous Underwater Vehicles(AUVs) Based on Potential Function and Behavior Rules," *2007 IEEE International Conference on Automation and Logistics,* pp. 569 - 573, 2007.

[14] H. Li, J. Peng, J. Xiao, F. Zhou, W. Liu and J. Wang, "Distributed Formation Control for a Cooperative Multi-agent System," *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou,* pp. 1893 - 1898, 2012.

[15] X. Ge and Q. Han, "Distributed Formation Control of Networked Multi-Agent Systems Using a Dynamic Event-Triggered Communication Mechanism," *IEEE Transactions on Industrial Electronics,* vol. 64, no. 10, pp. 8118 - 8127, 2017.

[16] A. l. Penga, S. Yang, G. Wen, A. Rahmani and Y. Yu, "Adaptive distributed formation control for multiple nonholonomic wheeled mobile robots," *Neurocomputing,* vol. 173, no. 3, pp. 1485 - 1494, 2016.

[17] C. Yoshioka and T. Namerikawa, "Formation Control of Nonholonomic Multi-Vehicle Systems based on Virtual Structure," *IFAC Proceedings Volumes,* vol. 41, no. 2, pp. 5149 - 5154, 2008.

[18] P. Lin and Y. Jia, "Distributed rotating formation control of multi-agent systems," *Systems & Control Letters,* vol. 59, no. 10, pp. 587 - 595, 2010.

[19] W. Ren and R. W. Beard, Distributed Consensus in Multi-vehicle Cooperative Control, Theory and Applications, Springer, 2007.

[20] C. Godsil and G. Royle, Algebraic Graph Theory - Graduate Texts in Mathematics, New York: Springer, 2004.

[21] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control,* vol. 49, no. 9, pp. 1465 - 1476, 2004.

[22] A. Jadbabaie, J. Lin and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control,* vol. 48, no. 6, pp. 988 - 1001, 2003.

[23] Z. Lin, M. Broucke and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control,* vol. 49, no. 4, pp. 622 - 629, 2004.

[24] R. O. Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control,* vol. 49, no. 9, pp. 1520 - 1533, 2004.

[25] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control,* vol. 50, no. 5, pp. 655 - 661, 2005.

[26] R. O. Saber and R. M. Murray, "Consensus Protocols for Networks of Dynamic Agents," *American Control Conference, 2003,* pp. 951 - 956, 2003.

[27] A. Svirin, "Basic Concepts of Stability Theory," math24.net, 2020. [Online]. Available: https://www.math24.net/stability-theory-basic-concepts/.

[28] G. Lafferriere, A. Williams, J. Caughman and J. J. P. Veerman, "Decentralized control of vehicle formations," *Systems and Control Letters,* vol. 54, no. 9, pp. 899 - 910, 2005.

[29] J.R.T. Lawton, R.W. Beard and B.J. Young, "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation,* vol. 19, no. 6, pp. 933 - 941, 2003.

[30] Z. Lin, B. Francis and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Transactions on Automatic Control,* vol. 50, no. 1, pp. 121 - 127, 2005.

[31] J. A. Marshalla, T. Fung, M. E. Broucke, G. M. T. D'Eleuterio and B. A. Francis, "Experiments in multirobot coordination," *Robotics and Autonomous Systems,* vol. 54, no. 3, pp. 265 - 275, 2006.

[32] M. Porfiria, D. G. Robersonb and D. J. Stilwell, "Tracking and formation control of multiple autonomous agents: A two-level consensus approach," *Automatica,* vol. 43, no. 8, p. 2007, 1318 - 1328.

[33] W. Ren, "Consensus strategies for cooperative control of vehicle formations," *IET Control Theory & Applications,* vol. 1, no. 2, pp. 505 - 512, 2007.

[34] P. Wang, "Navigation strategies for multiple autonomous mobile robots moving in formation," *Journal of Robotic Systems,* vol. 8, no. 2, pp. 177 - 195, 1991.

[35] P. Wang and F. Y. Hadaegh, "Coordination and control of multiple microspacecraft moving in formation," *Journal of the Astronautical Sciences,* vol. 44, no. 3, pp. 315 - 355, 1996.

[36] C. R. McInnes, "Autonomous ring formation for a planar constellation of satellites," *Journal of Guidance, Control, and Dynamics,* vol. 18, no. 5, pp. 1215 - 1217, 1995.

[37] M. A. Lewis and K.-H. Tan, "High precision formation control of mobile robots using virtual structures," *Autonomous Robots,* no. 4, pp. 387 - 403, 1997.

[38] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Transactions*

on Robotics and Automation, vol. 20, no. 5, pp. 865 - 875, 2004.

[39] T. Eren, P. N. Belhumeur and A. S. Morse, "Closing ranks in vehicle formations based on rigidity," in *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, NV, 2002.

[40] P. Ögren, E. Fiorelli and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Transactions on Automatic Control,* vol. 49, no. 8, pp. 1292 - 1302, 2004.

[41] K. D. Do, "Bounded controllers for formation stabilization of mobile agents with limited sensing ranges," *IEEE Transactions on Automatic Control,* vol. 52, no. 3, pp. 569 - 576, 2007.

[42] W. Ren and N. Sorensen, "Distributed coordination architecture for multi-robot formation control," *Robotics and Autonomous Systems,* vol. 56, no. 4, pp. 324 - 333, 2008.

[43] R. W. Beard and F. Y. Hadaegh, "Constellation templates: An approach to autonomous formation flying," in *World Automation Congress*, Anchorage, AK, 1998.

[44] R. W. Beard, J. Lawton and F. Y. Hadaegh, "A coordination architecture for spacecraft formation control," *IEEE Transactions on Control Systems Technology,* vol. 9, no. 6, pp. 777 - 790, 2001.

[45] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation,* vol. 14, no. 6, pp. 926 - 939, 1998.

[46] G. Díaz-García, L. Burbano, N. Quijano and L. F. Giraldo, "Distributed MPC and Potential Game Controller for Consensus in Multiple Differential-Drive Robots," in *2019 IEEE 4th Colombian Conference on Automatic Control (CCAC)*, Medellin, Colombia, 2019.

[47] S. Thrun, W. Burgard and D. Fox, Probabilistic robotics, Cambridge, Massachusetts: MIT Press, 2005.